INDOOR POSITIONING AND TRACKING METHODS FOR MOBILE WIRELESS DEVICES

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

José Luis Carrera Villacrés

von Ecuador

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik

Indoor Positioning and Tracking Methods for Mobile Wireless Devices

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

José Luis Carrera Villacrés

von Ecuador

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Der Dekan:

Bern, 7 October, 2019 Prof. Dr. Zoltan Balogh

Dedicated to my beloved family…

Dedicada a mi amada familia…

Acknowledgements

This work was carried out as part of my doctoral studies at the University of Bern. It would not have been possible without support of many people to whom I would like to express my acknowledgement and gratitude.
I would like to thank all members of the CDS group of the University of Bern for many fruitful discussions and support. Special thanks go to Prof. Torsten Braun for giving me the opportunity to join the Communications and Distributed Systems research group (CDS) of the University of Bern, and for all his support to the doctoral degree.
I would also like to thank Prof. Dr. Zan Li, for all his support, ideas, suggestions and motivation he has given me. Thank you for all the detailed comments and suggestions.
Furthermore, I must acknowledge my current and former colleagues of the CDS group who support me to get through the tough times. Special thanks go to Daniela Schroth for their kind help in all administrative work.
I would also like to express my gratitude to my parents, sister, and brothers. They were always supporting me and encouraging me to successfully complete my PhD studies. Despite the long physical distance, their wishes and thoughts are always with me.
I would like to express all my gratitude to my girlfriend Sophie Bongard, who has shared with me time of success but also hard time. Many thanks Sophie for supporting me even in the more difficult time.

Bern, 7 October 2019 José Luis Carrera Villacrés.

Abstract

The growing ubiquitousness of the Internet of Things (IoT) and the fast evolution of mobile computing have arisen new paradigms of context-aware applications where accurate and stable real-time indoor localization is an essential requirement. Indoor localization systems provide to mobile devices to locate themselves and provide the information to users. However, contrary to Global Positioning System (GPS) for outdoor environments, currently there is not a simple and accurate solution for indoor localization. Typically, indoor position-based services require higher localization accuracy than outdoor services. Additionally, the often limited computation and power resources on mobile devices put strong constraints on the algorithmic complexity design of solutions. Thus, indoor localization is still considered an open challenging problem.

The location-awareness will not only enable a vast amount of location-based services and applications such as intelligent transportation systems (ITS) and autonomous vehicles, but also support valuable location-aware communication enhancements such as proactive radio resource management, proactive handover optimization, etc. Thus, reliable localization methods become the underlying requirement to enable location aware-services. With the availability of more embedded sensors on mobile devices, numerous indoor localization techniques (i.e, ultrasonic, radio frequency tags (RFID), magnetic field, Wi-Fi, Bluetooth, etc.), have been proposed by exploring radio signals such as fingerprinting and radio-based ranging. However, the necessity to provide infrastructure able to manage complex computational processes and huge amount of data has arisen also. Cloud Computing (CC) enables high storage capacity and high computational processing power, network and computational overhead at the central cloud increases. This could lead to performance issues for applications where low latency is a fundamental requisite.

This thesis focuses on providing accurate and stable indoor localization. We work on terminal-based and network-based indoor positioning and tracking systems for

wireless mobile devices. To provide high and stable localization and tracking accuracy, we integrate radio signal information, inertial measurement information, and environmental information (e.g., earth's magnetic field, floor plan information) in probabilistic methods such as particle filters. The contributions in this thesis can be summarized as follows: First, enhanced methods to improve indoor localization and tracking accuracy. We address challenges for minimizing errors produced by inaccurate sensors and low sensing data rate on commodity mobile devices such as smartphones. Second, indoor localization with failure recovery methods. We introduce a localization failure method that is able to recover the system from localization failures, such as the kidnapping robot problem and global localization problem. The proposed failure recovery algorithms are based on machine learning models, which provide indoor zone level localization. We introduce two novel ensemble machine learning models, which are able to exploit physical information about the area of interest and information about the performance of individual machine learning models. Third, system architectures for indoor localization and tracking. To handle high complexity of the localization algorithms, we propose a Multi-Access Edge Computing architecture for indoor localization and tracking. Additionally, we introduce a novel reinforcement learning approach for robust and stable indoor localization and tracking.

Our proposed localization algorithms significantly outperforms the other commonly used range-based and fingerprinting-based positioning algorithms such as traditional particle filter, Kalman filter and trilateration algorithms.

Keywords: Indoor Positioning, Particle Filter, Reinforcement Learning, Internet of Things, Ensemble Learning Methods, Hidden Markov Model, Kidnapping Robot Problem.

Contents

List of Figures

1

Introduction

Awareness of the current position of a target enables a large range of context-aware applications. Context-aware applications must be able to timely react to the current physical context (i.e., environment) of mobile users. Thus, reliable localization becomes the underlying requirement of these applications. Although location information has become a transcendent part of the daily life of mobile users, currently there is not a simple and accurate solution for indoor localization. Additionally, the often limited computation and power resources on mobile devices put strong constraints on the algorithmic complexity design of solutions. Thus, indoor localization is still considered as an open challenging problem. We define localization accuracy as the error between the estimated position and the ground truth position of the target. Robustness is to guarantee the system reliability against localization failures. The development of such localization approaches includes several aspects including artificial intelligence methods and the choice of the system architecture. It is challenging for a localization approach to balance the strength and weakness of each design.

1.1 Overview

The current social and commercial importance of indoor location-based applications have attracted significant attentions in recent years. The location-awareness will not only enable a vast amount of location-based services and applications such as intelligent transportation systems (ITSs) and autonomous vehicles, but also support valuable location-aware communication enhancements such as proactive radio resource management, proactive handover optimization, etc. Thus, reliable localization methods become the underlying requirement to enable location aware-services.

The vision of indoor localization, however, entails big challenges. For example, the noise in low-cost Inertial Measurement Units (IMUs) on commodity mobile devices will introduce accumulated errors in the process of numeric integration during tracking [50]. Many wireless indoor positioning approaches have been proposed. Radio signals are most commonly used for indoor positioning, such as fingerprinting, radio-based ranging, etc. Radio-based indoor positioning has the intrinsic problem of signal unreliability due to multi-path propagation and non-line-of-sight (NLOS) signal conditions in indoor environments. One solution is to apply data-driven methods such as machine learning algorithms on received signal data, where a training dataset including measurements with known locations is used to train a model that estimates the location of a data sample. Wi-Fi received signal strength indicator (RSSI) is a common metric in both range-based and fingerprinting-based approaches. The earth magnetic field, which presents distortions over space due to the presence of ferromagnetic materials, is also used to improve localization accuracy [4]. However, to build up a fingerprint database is very time consuming. Typically, building a radio map database for fingerprints requires much more effort than range-based methods for localization.

Although several methods have been proposed to provide indoor localization, most of them implement running algorithms locally in the target mobile devices. However, the limited computational resources of mobile devices make it difficult to run complex algorithms. Real-time indoor localization requires significant computation, which is typically running on mobile devices with limited resources. Offloading the heavy computation to a third party server could resolve the problem, but the data transmission between the mobile devices and centralized server could lead to higher latency and unreliable performance, which is not tolerable for real-time applications.

1.2 Problem Statement

Several approaches have been proposed to provide indoor localization. Although some of them report high accuracy, there are some common problems and challenges to overcome in indoor localization systems. In this section, we describe these challenges and present the approaches that we target to solve them.

1.2.1 Indoor Localization and Tracking Accuracy

The vision of indoor localization and tracking on commodity mobile devices (e.g., smartphones), entails big challenges. For example, the IMUs on commodity smartphones are low-cost and have low power consumption but suffer from significant sensor errors. Low-cost IMUs on commodity devices will introduce accumulated errors in the process of numeric integration during tracking [50]. Thus, localization systems must consider additional information like Wi-Fi signals or floor plan information to deal with this kind of errors. However, Wi-Fi signals in smartphones tend to be unstable due to multipath effects, which are severe in indoor environments [113]. Moreover, observations show that Wi-Fi RSSI values often fluctuate over time even if the device is stationary. Unstable and vulnerable RSSI values introduce undesirable localization errors. Another issue is the achievable sampling rate of RSSI readings in Wi-Fi sensors. The sampling rate of inertial sensors can achieve 100Hz. However, the sampling frequency of Wi-Fi sensors is approximately 4Hz on commercial smartphones [50]. Therefore, inaccuracy of inertial sensors, instability, and low sampling frequency in Wi-Fi sensors introduce errors in the location estimation process. **Therefore, it is important to mitigate the tracking errors introduced by the inaccurate IMUs and low sampling frequency of Wi-Fi sensors in commodity mobile devices.**

1.2.2 Failure Recovery in Monte Carlo based Localization Systems

Monte Carlo Localization (MCL) is a probabilistic method that is widely used to design indoor localization systems. MCL is a sample-based method that represents the probability density of the mobile target position. There are some common problems to overcome in Monte Carlo based Localization systems (MCLS). Two of them are the global localization problem and the kidnapped-robot problem (KRP). The former happens when the localization system starts. Here, the initial position of the target is unknown. The target is located somewhere in the environment without any knowledge

of its position.

In contrast to the global localization problem, the KRP occurs during system operation. The convergence process of hypotheses (called particles in MCLS) causes the absence of samples in some zones (i.e., subareas) inside the area of interest. This leads to localization failures if the target is located in a zone without samples. This problem is a variant of the global localization problem [102]. Furthermore, sensor faults and mechanical problems can also lead to a condition similar to KRP. In this case, the KRP occurs when a well-localized target in operation moves to another location during a localization failure. Thus, the target is not aware of this change of position. The target device might firmly believe to be somewhere else at the time of the kidnapping.

With the problems mentioned above, most of the state-of-the-art localization approaches cannot be guaranteed never to fail [102]. Therefore, the ability to recover from failures is essential for truly autonomous MCLS. **Therefore, it is important to provide a method to guarantee system reliability against localization failures in MCLS.**

1.2.3 System Architectures for Indoor Localization Systems

Latency minimization is an essential requirement in real-time services. Although several methods have been proposed to provide indoor localization, most of them implement the running algorithms locally in the target mobile devices. However, the limited power supply and computational resources of mobile devices make it difficult to run complex algorithms. Moreover, the rising interest around the Internet of Things (IoT) and context-aware applications has introduced a variety of technologies to deal with all the produced data in the field of the IoT and context-aware applications [89]. Context-aware applications must be able to timely react to the current physical context (i.e., environment) of mobile users. Thus, real-time localization becomes the underlying requirement of these applications. However, real-time indoor localization requires significant computation, which is typically running on mobile devices with limited resources. Offloading heavy computation to a third party server could resolve the problem, but the data transmission between the mobile devices and centralized server could lead to higher latency and unreliable performance, which is not tolerable for real-time applications. **Therefore it is important to provide an architecture to meet application requirements of short latency and high resource-demanding to guarantee high indoor localization performance.**

1.3 Thesis Contributions

This thesis focuses on providing accurate, reliable, and stable indoor localization and tracking for wireless mobile devices. First, this thesis contributes with solutions to improve indoor localization accuracy in commodity mobile devices. We exploit an enhanced probabilistic approach to mitigate errors caused by unstable and low frequency rate sensors in commodity mobile devices. Our approach fuses ambient signals, inertial sensors and physical information of the environment to achieve high localization and tracking accuracy. Second, we propose an efficient approach to deal with localization failures problems, such as the global localization and the kidnapping-robot problem. Third, we also propose a network-based localization architecture to deal with limited computational resources of commodity mobile devices. We support our contributions in this thesis by theoretical analysis. Moreover, we evaluated in real scenarios prototype systems. In the following paragraphs, we summarize the main contributions in this thesis.

1.3.1 Enhanced Solutions for Improving Indoor Localization and Tracking Accuracy

The first contribution of this thesis is to improve indoor localization and tracking performance [20], [17], [19]. This work proposes efficient localization methods to deal with the instability and low sampling frequency of commodity sensors in mobile devices. Thus, we exploit an enhanced particle filter with asynchronous continuous correction phase to fuse ambient signals, inertial sensors, and physical information of the environment, to achieve high localization and tracking accuracy in complex indoor scenarios using commodity mobile devices such as smartphones. Details are summarized as follows:

- **Enhanced particle filter** to fuse range information estimated from Wi-Fi RSSI, IMUs and floor plan information for accurate indoor tracking.
- **Improved PDR methods** by considering magnetic field and angular velocity measurements to further improve the heading orientation estimation.
- **Asynchronous continuous correction phase** to mitigate the tracking errors caused by unstable Wi-Fi RSSI readings and low sampling frequency experimented in Wi-Fi sensors of commodity smartphones.

- **Efficient method to describe the physical environment** to further reduce the computation overhead on smartphones.

We conduct a set of extensive experiments to evaluate the aforementioned contributions. The terminal-based system has been implemented in a commodity smartphone. Experiments show that our tracking approach can achieve an average tracking error up to $1.15m$. The failure recovery method is able to detect a localization failure in $3.4s$ of average. It achieves a tracking mean error of $0.8m$. Thus, our indoor tracking approach achieves high accuracy and stable performance.

1.3.2 Automatic Failure Recovery Mechanism in MCLS

The second contribution of this thesis is to deal with localization failures problems in MCLS [17], [16]. We propose an efficient method to automatically recover the system from localization failures. Our recovery method uses machine learning approaches, which provide zone level localization [14, 21]. We incorporate zone level localization results in the data fusion method (particle filter) to faster recover the system from localization failures, such as the KRP and global localization problem. We define zones as subareas inside the target area (e.g., rooms, corridors). Since our proposed recovery failure method relies on zone level localization results, we also focus on providing enhanced machine learning methods to achieve high zone level localization performance. Details are summarized as follows:

- **Zone level localization methods based on enhanced learning models**. We propose two novel ensemble machine learning models to provide zone level localization. Both ensemble learning models combine conceptually different individual machine learning algorithms to predict class labels (i.e., rooms, landmarks). This combination is made by applying concepts of conditional probability and evidences about the prediction performance of individual predictors. However, the second proposed ensemble learning model also integrates coarse-grained floor plan information in a Hidden Markov Model (HMM) to achieve high zone level localization performance.
- **Failure recovery mechanism** to automatically recover the system from localization failures such as KRP and global localization failures. We integrate machine learning zone level localization results in the particle filter method to faster recover the system from localization failures.

We conduct experimental studies to evaluate our failure recovery mechanism and our ensemble learning models in complex office-like indoor environments. Experiment results show that our failure recovery mechanism can detect a localization failure in $3.4s$ of average, and the average time to recover from localization error is $34.2s$. Moreover, our proposed ensemble machine learning models are able to improve prediction accuracy up to 9.17% compared to traditional machine learning methods.

1.3.3 Network Architecture for Indoor Tracking Systems

The third contribution of this thesis is to deal with resource-intensive mobile applications such as indoor localization and tracking [15, 16]. We propose a MEC architecture where lightweight algorithms run on the mobile target devices with limit resources, whereas heavy calculations are offloaded to nearby edge servers. Thus, algorithmic complexity is not constrained by the limited computational resources of target mobiles devices. The MEC architecture brings the storage and processing capabilities close to the user. Thus, by providing an intermediate layer (i.e., MEC layer) at the network edge, MEC reduces significantly the network latency which is required for real-time applications [27]. Therefore, by distributed system architectures, we also focus on providing enhanced algorithms for indoor localization.

The localization algorithms are based on enhanced particle filters. Moreover, we propose a novel reinforcement learning-based approach for robust wireless indoor positioning. Our reinforcement learning-based approach assures the placement of samples (i.e., particles) over areas where the desired distribution is large (i.e., areas with high probability of containing the ground truth position). This scheme reduces convergence time and provides autonomy and robustness to the system. To deal with multi-path effects that detrimentally affect range-based localization approaches, we provide an effective method to infer LOS and NLOS conditions. Ensemble machine learning methods (provided also in this thesis), support to choose the proper ranging models that are specific for each zone (i.e., are where the target is located). If the mobile target (i.e., object to be located) and a ranging Anchor Node (AN) are at the same zone, the system adopts LOS ranging models concerning this AN. Furthermore, ranges to ANs located at different zones than the mobile client are calculated by NLOS ranging models. Contribution details are summarized as follows:

- We design a **particle filter-based reinforcement learning algorithm** for robust

wireless indoor positioning system. The particle filter fuses ensemble learning models for zone localization, radio-based ranges, IMUs, and coarse-grained floor plan information to achieve accurate and stable indoor tracking performance. In the particle filter, we provide a reinforcement learning-based resampling method to guarantee system reliability against localization failures.

- We propose a **distributed ML-based network architecture** for indoor tracking, where lightweight ML algorithms (indoor zone prediction) are running on the mobile devices with limited resources and heavy ML calculations are offloaded to nearby edge servers to support complex and heavy calculations.
- We propose a **ranging model to deal with multi-path effects**. The ranging model is based on zone localization results. The zone prediction method supports to choose the proper ranging models that are specific for each zone. If the mobile client (i.e., object to be located) and a ranging Anchor Node (AN) are at the same zone, the system adopts LOS ranging models with respect to this AN.

We perform a set of experiments in complex office-like environments along different moving paths to validate the performance and reliability of our system. With an average tracking error of 0.97 meters and failure recovery time latency of 1.5 seconds, our proposed localization method overcomes traditional solutions in terms of reliability, stability, and accuracy.

1.4 Thesis Outline

The remaining of the thesis is structured as follows. In Chapter 2, we review the theoretical background and related work to fully understand the underlying basis of the approaches proposed in this thesis. Then, our main contributions are structured in two parts: Part I (Chapter 3, 4) introduces our work on enhanced methods to improve localization and tracking performance with localization failure recovery. Part II (Chapter 5, 6) proposes distributed system architectures, where lightweight algorithms run on the mobile target devices, whereas heavy calculations are offloaded to nearby edge servers. Thus, in Part II, we extend the localization approaches presented in Chapters 3 and 4 to a distributed system architecture. In the following paragraphs, we summarize the contributions included in each Chapter.

In Chapter 3, we propose an enhanced particle filter to fuse range information es-

timated from RSSI, IMUs as well as floor plan information for indoor tracking. We propose an asynchronous continuous correction phase to mitigate the tracking errors caused by unstable RSSI readings and low sampling frequency experimented in Wi-Fi sensors of commodity smartphones. we include an enhanced PDR method by considering magnetic field and angular velocity measurements to further improve the heading orientation estimation. Moreover, we propose an efficient discretized graph-based method to describe the physical environment. The proposed algorithms are designed in a terminal-based system, which consists of commercial smartphones and Wi-Fi access points.

In Chapter 4, we focus on localization failures recovery methods to achieve robust localization performance. We propose an efficient method to recover the localization system from localization failures. Our recovery method relies on machine learning techniques, which provide zone level localization. The fingerprint database is built by taking fingerprint measurements while walking randomly through the environment, which requires only zone-labeled samples. Thus, the fingerprint database is built in a short period. Since our proposed failure recovery method relies on zone localization, we also focus on advanced machine learning methods to improve zone level localization performance. We propose two novel ensemble learning methods, which combine conceptually different individual machine learning algorithms to predict class labels (i.e., rooms, zones, landmarks). The first proposed ensemble learning method applies concepts of conditional probability and evidences about the prediction performance of individual predictors. The second proposed ensemble learning method relies on Hidden Markov models and coarse-grained floor plan information.

In Chapter 5, we exploit a distributed system architecture to design a particle filter-based reinforcement learning algorithm for indoor tracking with failure recovery. Our approach fuses zone localization results, radio-based ranges, IMUs, and coarse-grained floor plan information to achieve accurate and stable indoor tracking performance. We provide a reinforcement learning-based resampling method to guarantee system robustness against localization failures. Moreover, the reinforcement learning approach makes the localization system converge much faster than other particle-based localization methods.

In Chapter 6, we provide a MEC-based architecture to localize mobile wireless devices in indoor scenarios. We also provide a probabilistic method to integrate room prediction results, radio frequency, and environmental information for accurate indoor

localization. Since the computational complexity is offloaded to a MEC server, the localization algorithms are not constrained by the limited computational resources of mobile devices.

Finally, Chapter 7 concludes the thesis by summarizing the contributions of this work.

2

Theoretical Background and Related Work

2.1 Introduction

Providing indoor localization has acquired special attention due to the growing number of context-aware applications and the current importance of IoT. Although several indoor localization approaches have been proposed, there is not a single solution considered as a standard for indoor localization. Thus, indoor localization remains a challenging research problem. In this chapter, we review the theoretical background and related work to the approaches proposed in this thesis.

2.2 Communication Networks and Architectures

2.2.1 IEEE 802.15.4

The IEEE 802.15.4 standard specifies the PHY and MAC layers of the low rate Wireless Personal Area Network (WPAN). It is used in applications that require low data rates and low power consumption.

The IEEE 802.15.4 standard defines the protocol and compatible interconnection for devices using low-data-rate, low-power, and low-complexity short-range radio frequency transmissions [2]. This standard describes the physical layer (PHY) and medium access control (MAC) layer specifications for wireless connectivity with fixed, portable, and moving devices with no battery or limited battery consumption requirements. It is widely used in wireless sensor networks, home automation, and industry applications. Furthermore, the standard provides modes to enable accurate ranging. This standard uses extremely large frequency ranges and low transmit power. Thanks to its robustness against interference and short pulse duration, it is also suitable for accurate indoor positioning. The IEEE 802.15.4 architecture is defined in terms of layers. Each layer is in charge of some parts of the standard and provides services to the higher layers.

An IEEE802.15.4 device comprises at least one PHY and a MAC layer. The PHY layer contains the radio frequency (RF) transceiver. The MAC layer provides access to the physical channel for all types of transfer.

IEEE 802.15.4a

The IEEE 802.15.4a standard provides two alternate PHY layers for low rate WPAN: Chirp Spread Spectrum PHY (CSS-PHY) and Ultra Wide Band PHY (UWB-PHY). Since the IEEE802.15.4a UWB-PHY standard is used in our work, we include in this section only the IEEE802.15.4a UWB-PHY layer and some parts of the IEEE802.15.4 MAC layer specifications. Further details about this standard can be found in [2]. IEEE 802.15.4a UWB-PHY is used for applications that require high precision ranging and low power transmission. Hereinafter IEEE 802.15.4a UWB-PHY is referred as UWB.

UWB is a high-speed and short-range radio technology for wireless communication. UWB technology has many potential advantages, including the delivery of high

Table 2.1: UWB Strengths and Weakness

Strengths	Weakness
High noise immunity	Long signal acquisition times
Low power	Limited emission requirements
High material penetration	Interference with other radio technologies
High immunity to multipath fading	High cost
Potentially high data rates	Short range (up to 30m)

throughput by sharing a large amount of spectrum with other UWB devices. However, for indoor localization, the main advantage of UWB is its robustness against multipath effects.

UWB transmits ultra short-pulses with time period less than 1 nanosecond over a large bandwidth in the frequency range from 3.1 to 10.6GHz. UWB uses a very low duty cycle which results in reduced power consumption. Thus, due to its drastically different signal type and radio spectrum, UWB technology is highly immune to interference from other radio signals. The low frequencies included in the broad range of the UWB spectrum can penetrate a variety of materials, including walls. Moreover, the very short duration of UWB pulses makes UWB less sensitive to multipath effects. This allows the identification of the main path in the presence of multipath signals.

UWB technology was primarily designed for short-range communication. However, due to its robustness against multipath effects, its high ability to penetrate materials, and its immunity to interference from other radio signals, UWB is commonly used in distance estimation, localization and tracking through Time of flight (TOF). In TOF, a radio wave signal is sent from a transmitter to a receiver device and back. Then, the distance between the transmitter and receiver devices is estimated by multiplying the time of flight of the wave with the speed of radio waves. Table 2.1 summarizes some strengths and weaknesses of the UWB technology.

The UWB is in charge of the activation and deactivation of the radio transceiver, energy detection, link quality indication, channel selection, ranging, transmitting and receiving packets across the physical medium. There are three different bands groups: low-band, high-band, and sub GHz band. These groups include 16 channels with 499.2MHz. The main characteristics of UWB are shown in Table 2.2. Figure 2.1 depicts the data frame structure of UWB.

The preamble code is used to build symbols of the SYNC portion synchronisation

Table 2.2: UWB PHY Main Characteristics

Characteristic	Description
Bandwidth	499.2 MHz, 1081.6 MHz, 1331.1 MHz and 1354.9 MHz
Frequency channel	Channel 3, 7, 11 and 15
Data rate	0.11 Mbps, 0.85 Mbps, 6.8 Mbps and 27.4 Mbps
Centre frequency	4492.8 MHz, 7488.0 MHz and 9984.0 MHz
Forward error correction	Reed Solomon coder

Table 2.3: Preamble Code Sequences UWB PHY (31 length)

Index	Sequence	CH number
1	-0000+0-0+++0+-000+-+++00-+0-00	0,1,8,12
2	0+0+-0+0+000-++0-+—00+00++000	0,1,8,12
3	-+0++000-+-++00++0+00-0000-0+0-	2,5,9,13
4	0000+-00-00-++++0+-+000+0-0++0-	2,5,9,13
5	-0+-00+++-+000-+0+++0-0+0000-00	3,6,10,14
6	++00+00—+-0++-000+0+0-+0+0000	3,6,10,14
7	+0000+-0+0+00+000+0++—0-+00-+	4,7,11,15
8	0+00-0-0++0000–+00-+0++-++0+00	4,7,11,15

header (SHR) preamble. UWB supports 31 and 127 length code sequences of preamble code. The length 31 code sequences is shown in Table 2.3.

IEEE802.15.4 MAC Layer

The MAC layer provides beacon management, channel access, frame validation, acknowledged frame delivery, association, and disassociation. Moreover, the MAC layer is in charge of implementing application security mechanisms.

Frame Structure

The frame structure allows robust transmission on noisy channels while keeping low complexity in the frame design. Figure 2.1 shows the frame structure of the IEEE802.15.4 standard. The MAC frame is passed to the PHY layer as a PHY service data unit(PSDU) which becomes the PHY payload.

Radio Frequency (RF) technology is one of the promising solutions to provide real-time indoor localization. Compared to other RF technologies, ultra wide band (UWB) has received increased interests due to its capability to reduce the localization errors

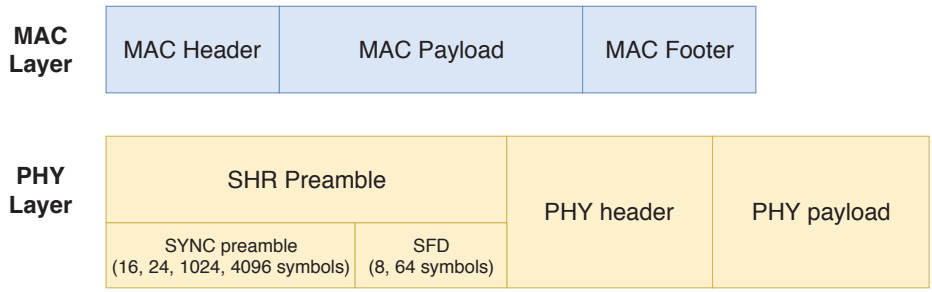


Figure 2.1: IEEE802.15.4a frame structure

to lower than one meter. UWB is robust to multi-path effects because UWB radios are able to differentiate pulses reflected from different objects. Thus, UWB-based localization achieves better accuracy and reliability than other wireless technologies such as Wi-Fi or Bluetooth, which normally achieves localization accuracy of several meters [85].

2.2.2 IEEE 802.11

Wi-Fi describes a local wireless network that uses radio waves to transmit data. There are several versions of Wi-Fi as defined in the IEEE specifications, including 5 GHz, and 2.4 GHz frequency. Certain versions of Wi-Fi achieve a transmission speed up to several gigabits per second. Thus, Wi-Fi is typically used to transfer large amounts of information. Table 2.4 summarizes some strengths and weakness of the Wi-Fi technology.

Table 2.4: Wi-Fi Strengths and Weakness

Strengths	**Weakness**
Large range (up to 150 m)	Prone to interference
Widely available	Requires security protocols
Network scalability	Prone to denial of service
Client mobility	Highly affected by multi-path fading

The IEEE 802.11 standard [1] is a set of physical (PHY) and Medium Access Control (MAC) layer standards for implementing wireless local area network computer communication (WLAN). This standard is the most widely used wireless computer networking standard. The IEEE 802.11 standard operates in the Industrial Scientific, and Medical (ISM) band. The IEEE 802.11 standard is mainly used to provide networking capabilities and connect devices in public, private, and commercial networks. Initially,

WiFi had a range of coverage of about 100 meters. However, this range has increased to about 1 kilometer (km) in IEEE 802.11ah to be optimized for IoT services [22].

Most of the modern mobile devices, such as laptops and smartphones enable Wi-Fi connectivity. Thus, Wi-Fi technology has become an ideal candidate for indoor localization and one of the most commonly studied technologies for indoor localization. Because existing Wi-Fi access points can be used as reference anchor nodes for signal collection, localization systems can be built without additional infrastructure.

The 802.11 standard is a set of over-the-air modulation techniques. This standard provides the basis for several wireless network applications. The IEEE 802.11 standard also defines the radio frequency spectrum in each country over the world.

The 802.11n enhancements allow receiving and/or transmit simultaneously by using multiple antennas. The 802.11n defines many to many antenna configurations (MxN), ranging from 1×1 to 4×4. This transmission technology is called MIMO. MIMO uses Spatial Division Multiplexing as channel access method. It allows multiple independent data streams transferred simultaneously within one spectral channel of bandwidth. Thus, MIMO increases the data throughput as by increasing the number of resolved spatial data streams. Since part of our work is focused to IEEE 802.11n-based network systems, we review some relevant knowledge of this standard in the following paragraphs.

IEEE 802.11n Medium Access Control Layer (MAC)

In IEEE 802.11n protocol, the MAC layer manages the access to the shared physical media (i.e., air interface). Thus, the MAC layer enables Wi-Fi devices in range to communicate effectively. The MAC layer takes data from the higher sub-layer called Data Link layer (LLC), adds header and tail bytes, and sends the data to lower Physical layer (PHY). The reverse process occurs when receiving data from the PHY layer.

Received Signal Strength Indicator (RSSI) is a MAC layer component that measures the transmitted signal power by characterizing the attenuation of radio signals during propagation. RSSI is accessible in wireless techniques from UWB, ZigBee, and Wi-Fi to cellular networks.

The MAC layer implements Medium Access Control mechanisms to control access to a shared medium. Thus, the MAC layer allows multiple devices to reliably communicate

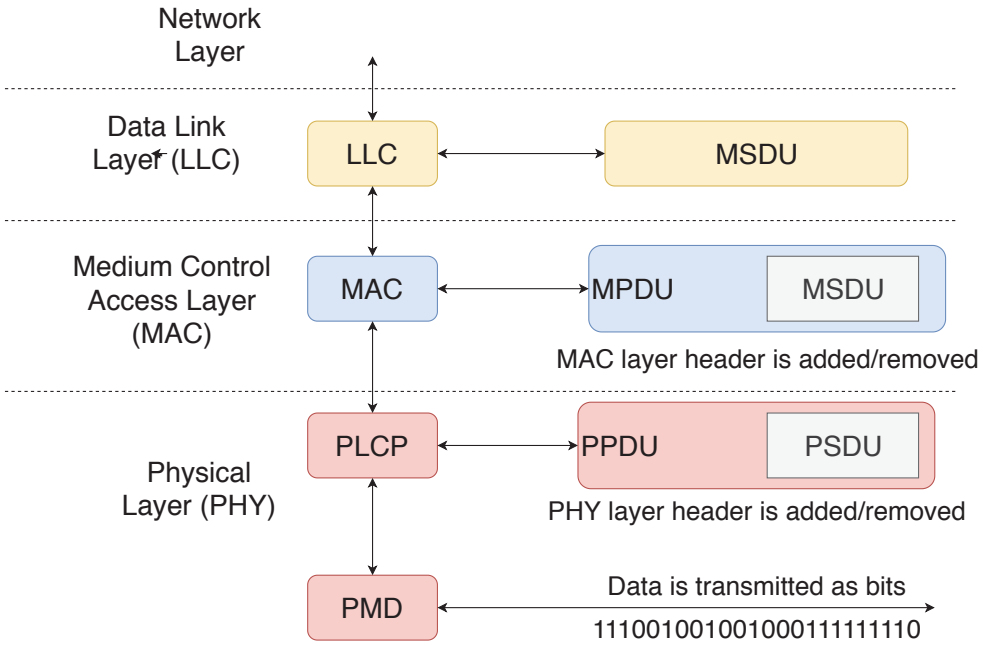


Figure 2.2: MAC layer functions and interactions. MAC layer is a sub-layer between LLC and PHY layer.

by sharing the medium as specified in the standard.

MAC layer implements processes for scanning, authentication, association, power saving and fragmentation. Figure 2.2 shows the functions of the MAC layer and its interaction with the PHY and the LLC layers.

The MAC layer takes a packet of data called MAC Service Data Unit (MSDU) from Logical Link Control (LLC) sub-layer. MAC adds necessary header and tail bytes to form MAC Protocol Data Unit (MPDU). MPDU is then sent to the physical layer for transmission. In the physical layer, the Physical Layer Convergence Protocol (PLCP) appends a PHY-specific preamble and header fields to the MPDU that contain information needed by the Physical layer transmitters and receivers. Physical Medium Dependent (PMD) sublayer provides transmission and reception of Physical layer data units between two stations via the wireless medium. The reverse process is performed when MAC receives a packet from the PHY layer.

The scanning process is performed in a passive and active fashion. In the passive scanning, the station looks for beacon frames that are regularly sent by an Access Point (AP). These frames contain information describing the network. To connect to an AP,

the station selects the AP with the strongest signal if multiple APs are located within range. the station then attempts to connect to the strongest AP. The communication is performed in the channel in which the AP is operating.

In the active scanning process, the station sends a request either to a specific AP or to any AP within range. It expects a response from one or more APs. Then, the station selects the strongest AP and attempts to connect.

The MAC layer implements the network multiple access method called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA consists of two sub-processes: Carrier Sensing and the Collision Avoidance. In the Carrier Sensing sub process, Wi-Fi stations sense the media interface. It the channel is idle, stations are allowed to transmit. In the Collision Avoidance process, the station senses if the channel is busy, then it waits for a random interval before transmitting. The randomness reduces the probability of collisions among stations transmitting at the same time. Even though the randomness, collisions can still occur. However, they are inferred when no acknowledgment message (ACK) is received by the sender. If ACK is not received, station backs off for a random duration and repeats the process.

2.2.3 The Internet of Things Network

According to CISCO systems, the Internet of Things (IoT) network can be defined as "a pervasive and ubiquitous network which enables monitoring and control of the physical environment by collecting, processing, and analyzing the data generated by sensors or smart objects" [100]. Thus, IoT is a network that connects physical devices, sensors, vehicles, electronic objects, etc. to the Internet. Those devices embedded software, actuators, and sensors to collect and exchange data. However, the IoT functionalities go beyond this interaction by offering connections and networking services between such as transport services, community services, etc. [74]. It is a network that connects everything to the Internet to exchange information and communication through devices with agreed protocols. Interconnected devices are able to be identified, located, and they are able to monitor and manage things connected to the network [25]. Therefore, the Internet is no longer bound by the desktop but goes out into the world of other things [74]. Figure 2.3 shows the common IoT architecture defined in [100].

The underlying technologies that lead to the IoT concepts have existed for some time

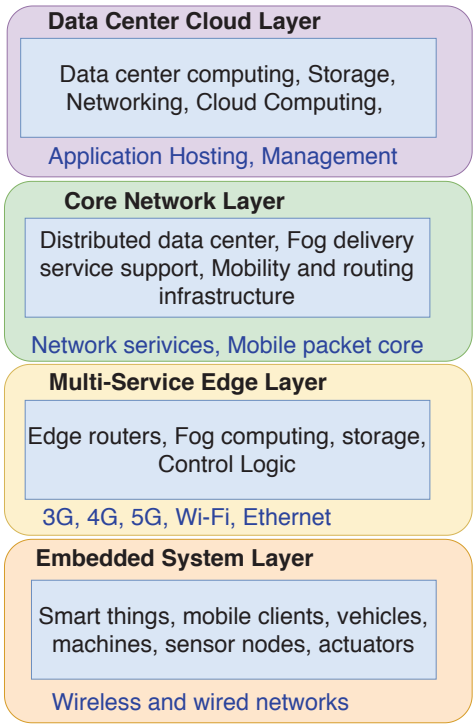


Figure 2.3: The common IoT architecture.

(e.g., Machine-to-Machine (M2M) communications, Radio Frequency Identification (RFID), Location-Based Services (LBS), Lab-on-a-Chip (LOC) sensors, Augmented Reality (AR), robotics, vehicle telematics, etc.). The common feature of these technologies is that they combine embedded sensory objects with communication intelligence, running data over a mix of wired and wireless networks [100].

Because of the huge amount of data generated by the interaction process in IoT, cloud technology suits IoT applications. Moving IoT application data to the cloud can reduce the cost and complexity related to hardware management. Thus, it is needed to aggregate several IoT device messages using mobile cloud computing closer to the device users to improve latency and response time. IoT devices are connected over different communication technologies such as 3G, 4G, 5G, Wi-Fi, UWB. In general, the data transmissions are small messages, encrypted and come in different forms of protocols such as, MQTT, CoAP. Therefore, there is a requirement for a low latency aggregation point to manage the heterogeneous protocols and data processing of analytic from data collected from different IoT applications. The edge server has the capability to resolve those challenges.

There are several definitions about IoT architecture in the literature. Here, we include a high level architecture's description provided by [100]. Further definitions can be found in [13, 100, 104].

Embedded Systems Layer

The Embedded Systems Layer is the first layer of the IoT architecture. It is comprised of embedded systems, sensors and actuators. Sensor nodes and actuators are small devices, usually with limited processing and power and capabilities. These devices could be deployed in remote locations.

The hardware of a sensor node and an actuator often includes four parts: the power management module, a sensor, a microcontroller, and a wireless transceiver. The power module provides a reliable power source for operating the system. These devices are in charge of collecting and transforming environmental signals, such as light, humidity, etc into electrical signals and then transferring them to the microcontroller for further processing.

Multi-Service Edge Layer

This layer supports wired and wireless communications. Thus, this layer must implement many communication protocols (e.g., Zigbee, IEEE 802.11, 3G, 4G, and 5G). This layer must be modular to scale to accomplish growth requirements. The applications and services implemented in this layer should be similar so that additional applications and services can be easily added.

Core Network Layer

The Core Network layer is in charge of providing paths to carry and exchange data and network information between multiple networks. This layer is able to manage IoT traffic and data. The main difference between IoT Core Network Layer and conventional core layers is that in IoT the traffic and data may differ, for instance, variable packet size.

Data Center Cloud Layer

The Data Center Cloud Layer is in charge of providing hosting for applications and services to manage end-to-end IoT architecture. This layer is a pool of computational resources which are interconnected by a communication network.

2.2.4 Multi-Access Edge Computing (MEC)

Multi-Access Edge Computing (MEC) is a cloud-based Information Technology (IT) service environment located at the edge of the network [23]. Figure 2.4 shows the basic MEC architecture. The MEC servers are closer to the end user than cloud servers. Therefore, even though the MEC servers have less computation power than the cloud servers, they are still able to provide better QoS and lower latency to the end users. MEC is an emerging technology designed to provide real-time processing, high-bandwidth, low-latency access by reducing the network stress [8]. MEC servers are usually deployed close to the Radio Network Controller (RNC), and allow for delay-sensitive and context-aware applications to be executed in close proximity to end users [105]. Thus, MEC adapts cloud computing to the mobile environment where data are stored and processed outside mobile devices.

MEC Framework and Architecture

The European Telecommunications Standards Institute Industry Specification Group (ETSI ISG) [35], has published the specifications on the framework and reference architecture for MEC technology [34]. In the following paragraphs, we summarize these specifications.

MEC Reference Framework

High level functional entities are involved in the reference MEC framework. These entities are grouped in three layers: network-layer, host-layer, and system-layer.

The network-layer is related to external networks such as local networks, third Generation cellular network (3GPP), and the external networks (e.g., Internet). The host-layer includes the Mobile Edge (ME) host. In the ME host, the ME platform, the ME applications, and the ME virtualization are performed. The system-layer has the overall visibility of the ME system [88].

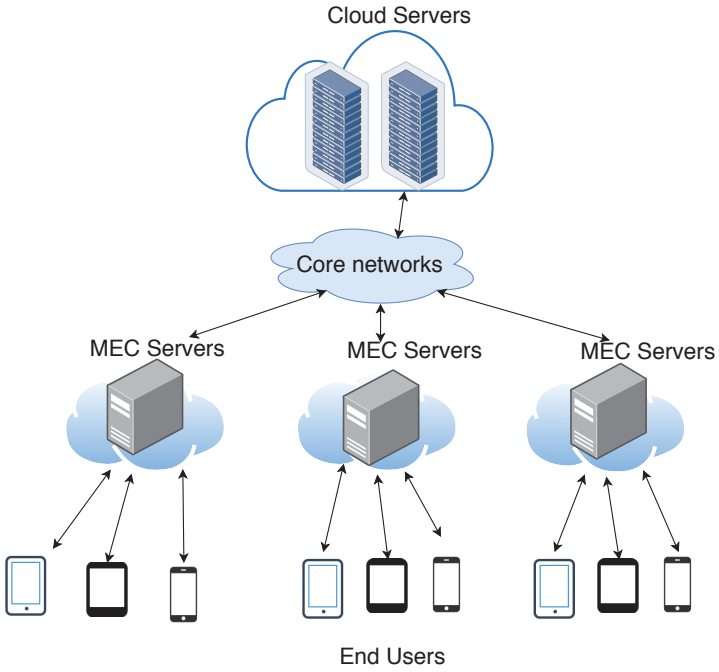


Figure 2.4: Graphical Representation of a Basic MEC architecture.

MEC Reference Architecture

As the network layer is not visible in the MEC reference architecture, entities are grouped in two layers: Host layer and System layer. The Host layer includes the MEC host. Through MEC platform and Virtualization platform, the MEC host provides processing, data storage and network resources for MEC applications. The virtualization infrastructure performs the forwarding rules received by the MEC platform and routes the traffic among the MEC applications. The MEC platform receives the traffic forwarding rules from the MEC platform manager.

MEC applications run as virtual machines on top of the virtualization infrastructure. The MEC applications interact with the MEC platform to consume services provided by the MEC platform. However, MEC applications can provide services to the MEC platform also. MEC applications must provide information related to resources, allowed latencies. Then, MEC host assignment is done by evaluating these requirements.

The MEC platform manager handles the application initialization and termination. Moreover, the MEC platform manager provides information to the MEC orchestrator

related applications events.

MEC orchestrator (MEO) has a view over the resources of the entire MEC network. Thus, MEO has information related to the entire system. The MEO maintain information about the deployed MEC hosts, available services and resources, initialized applications and the topology of the network. Moreover, the MEO manages the initialization of applications by providing instructions to the Virtualization Infrastructure.

The virtualization infrastructure manager (VIM) controls the virtualization resources of the MEC hosts. The Operation Support System (OSS) is in charge of running the MEC applications in the desired location of the network. Thus the OSS manages the requests to initialization and terminates the ME applications from the clients in the user equipment (UE) and the customer-facing service (CFS) portal. The CFS portal provides business-related information for users.

2.3 Inertial Sensor and Magnetometer based Localization

2.3.1 Inertial Sensors

Inertial sensors are micro-electromechanical (MEM) devices based on inertia. Inertial sensors measure force, and angular rate of a body, using a combination of sensors such as accelerometers, gyroscopes. Magnetometers are not based on inertia. However, magnetometers are commonly embedded together with the inertial sensors on mobile devices. Magnetometers measure magnetic fields. The raw data obtained from inertial sensors is often referred as Inertial Measurement Units(IMUs).

Accelerometer

An accelerometer is an electromechanical device that measures the physical acceleration forces experienced by a body. Such forces could be static (e.g., gravity), or dynamic to sense movement or vibrations of a specific body. Some sensing technologies also make use of some physical parameters from the environment, such as temperature, pressure, force, and light to improve acceleromenter reading accuracy. Typical accelerometers are multiple three-axis devices. Most of the modern smartphones embedded three-axis models, whereas cars simply use only a two-axis

to determine the moment of impact [42].

Gyroscope

Gyroscope sensors are devices that measure the angular velocity of a body. Gyroscopes are also known as angular rate sensors or angular velocity sensors. The gyroscope measures rotational velocity along the axis X, Y and Z also referred as Roll, Pitch, and Yaw. [86]. This rotational velocity can be expressed in degrees over seconds.

Magnetometer

A magnetometer is a device that measures magnetism, either the magnetization of magnetic material or the relative change of a magnetic field at a particular location. The earth's magnetic field can be measured with by a magnetometer. The magnetometer sensors in modern smartphones use a Hall-effect sensor to measure the magnitude of a magnetic field. The Hall-effect sensor produces a voltage which is directly proportional to the magnetic field strength. Then, the voltage pattern is used to measure the magnetic field intensity. Since a magnetometer is able to detect the earth's magnetic field, the magnetometer is a fundamental component to determine the relative orientation of a body relative to the Earth's magnetic north [87].

2.3.2 Pedestrian Dead Reckoning

Pedestrian Dead Reckoning (PDR) is an useful component for localization and tracking. Inertial sensors can be adopted to implement pedestrian movement detection such as step recognition, stride length estimation, and heading orientation estimation. PDR systems derive the new location based on the previously determined location by using inertial sensor readings (IMU). There are several works that have adopted pedestrian dead reckoning (PDR) as a fundamental component for indoor tracking.

Due to the fast development of modern smartphones, PDR relying on IMUs has attracted increasing research interests. Basically, PDR systems derive the new location based on the previously determined location by using sensor readings. Inertial sensors can be adopted to implement pedestrian movement detection such as step recognition, stride length estimation, and heading orientation estimation. For instance, the authors of [10,36] determined the heading orientation based on gyroscope

measurements, whereas the displacement is estimated from accelerometer readings. In [10], authors defined a method named Heuristic Drift Elimination (HDE), which is intended to deal with the accumulated errors. However, HDE requires specialized sensors deployed on the foot of the pedestrian. The authors of [54] adopted magnetic field sensor and accelerometer readings to estimate the heading orientation. In this, work authors defined a walking and running model based on accelerometer measurements. In [3], authors used readings of the gyroscope to identify physical turns of the pedestrian user, whereas the walking distance is determined by readings of the accelerometer. PDR systems measure position changes rather than the absolute position, which results in an accumulation of sensor errors over time. To deal with cumulative errors, PDR methods presented in this thesis consider additional information like Wi-Fi signals and floor plan information. Moreover, we incorporate a magnetometer and gyroscope filters to further smooth the heading orientation errors produced by magnetic field interference and noise in low-cost sensors on commodity mobile devices.

2.4 Signal based Localization Methods

Indoor environments are rich in different types of ambient signals. We consider ambient signal any electrical impulse, radio wave, or force from the surrounding environment that can be measured by sensor devices. Because of its ubiquitous availability, indoor ambient signals are often used in indoor positioning. Different parameters of indoor signals can be used to locate the targets, such as RSSI, time information and magnetic field distortions. Indoor signal-based indoor localization can be classified as range-free and range-based methods.

2.4.1 Range-based Localization

Range-based approaches convert the measured radio signal parameters into range values, which indicate the distance between the target mobile device and radio transceiver. This process is called ranging. After ranging, different positioning algorithms can be used to estimate the absolute locations of the targets, such as trilateration and multilateration [59]. Trilateration and multilateration determine the position of the target based on the distances to some anchor nodes. However, ranging accuracy is detrimentally affected by multi-path effects especially in Non-Line of

Sight (NLOS) conditions. To reduce the negative influence of multi-path effects, NLOS conditions need to be identified, and then some methods (e.g., adaptive propagation models) can be adopted to mitigate its negative influence.

Range-based localization approaches use radio propagation models to define the relationship between the propagation distances and radio parameters. Thus, ranges can be derived by using signal parameters such as RSSI or arrival time of radio signals. In theory, RSSI monotonically decreases with increasing propagation distance [59]. Some models to relate RSSI to the propagation distance have been proposed e.g., Log Distance Path Loss (LDPL) [93], Nonlinear Regression Model (NLR) [59]. These models are called propagation models.

The propagation model that describes the LDPL can be described by Equation 2.1:

$$P(r) = P(r_0) - 10 \cdot \gamma \cdot \log_{10}(\frac{r}{r_0}), \tag{2.1}$$

where $P(r)$ is the received signal power at certain distance r. $P(r_0)$ refers to the power loss in a free space. γ is the path loss efficient. Therefore, r can be calculated as follows:

$$r = 10^{(\frac{P(r_0)-P(r)}{10\cdot\gamma})}, \tag{2.2}$$

The NLR model is defined by Equation 2.3:

$$r = \alpha \cdot e^{(\beta \cdot P(r))}, \tag{2.3}$$

where r is the distance between the transmitter and the receiver. Both α and β are environmental variables, which values are determined by a training process. In indoor environments, the accuracy of the propagation models is affected by multipath and Non Line of Sight (NLOS) propagation [59].

There are some proposed time-based ranging methods to determine the distance between a transmitter and a receiver. Time-based ranging methods rely on precise time-stamping of incoming and outgoing messages [82]. For instance, in one way ranging methods, messages are not exchanged bidirectional (i.e., between transmitter and receiver) but unidirectionally. Therefore, synchronization methods between transmitter and receiver must be implemented. Since ranging messages are bidirectional in two way ranging methods, synchronization between transmitter and receiver is not required [94]. As shown in Figure 2.5 Tx transmits a ranging message to Rx. Then, the

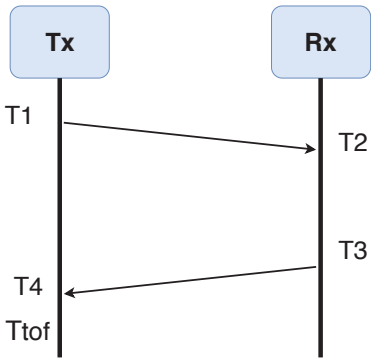


Figure 2.5: Two way ranging methods.

distance between Tx and Rx is derived from the propagation time called time of arrival T_{tof} is derived from T_{tof} and the signal propagation speed. Since the clocks in Tx and Rx are not synchronized, an additional message is transmitted. The time-of-flight T_{tof} can be calculated as follows:

$$T_{tof} = \frac{(T_4 - T_1) - (T_3 - T_2)}{2}, \tag{2.4}$$

The distance d between Tx and Rx is calculated as follows:

$$d = T_{tof} \cdot S_{rs}, \tag{2.5}$$

where S_{rs} is the radio signal propagation speed. As shown in Figure 2.5, two way ranging method requires to exchange two messages between Tx and Rx.

In complex indoor environments, Wi-Fi signals suffer from some random variations during transmission. This random behaviour is produced by the presence of multiple obstacles such as walls, furniture, ceiling, etc. Obstacles introduce a mixed transmission between Line of Sight (LOS) and Non-LOS (NLOS) conditions. NLOS propagation induces significant bias in power-based ranging [112]. Hence, the main challenge facing range-based localization approaches is to have a suitable and accurate propagation model to estimate ranges accurately with respect to actual distance. Many of proposed range-based solutions rely on having precise knowledge of mobile target device radio settings [29], [45] [64]. However, in real scenarios, this information is not available neither at the user side nor at the network side. Thus, random signal variations during transmissions make the distance estimations vary in unpredictable ways, which leads to introduce errors in range estimation processes. To deal with this

negative influence of random signal variations effects, NLOS conditions need to be identified, then some methods can be adopted. In this thesis, we exploit zone level localization to choose the proper ranging models that are specific for each zone. If the mobile device and a ranging Anchor Node are at the same zone, the system adopts LOS ranging models with respect to this AN.

2.4.2 Range-free Localization

Fingerprinting [7] is the most common range-free method. It is often used because of its robustness to multipath propagation. Fingerprinting-based indoor localization systems usually consist of two phases: training phase (off-line) and localization phase (on-line). In the off-line phase, the fingerprint database is built by collecting various types of radio signals in the target indoor environments. The earth magnetic field (MF) in indoor environments presents distortions over space due to the presence of ferromagnetic materials. These MF distortion patterns can also be used to identify indoor locations [4]. MF and RF observations can be used as fingerprints to detect unique locations in indoor environments. However, it is very time consuming to build up a radio map, which is required to locate the targets in fingerprinting. In the on-line phase, the observed fingerprint at an unknown location is compared with the stored fingerprints in the fingerprint database to determine the closest match. Any single learning model can be applied. However, ensemble learning models usually allow better predictive performance compared to single models [60]. Fingerprinting-based methods that build the classification model exclusively based on previously observed data (i.e., fingerprint database) are called discriminative learning methods.

Various machine learning-based approaches have been proposed that use fingerprinting to estimate user indoor locations. Machine learning-based indoor localization can be classified into generative or discriminative methods, which builds the machine learning model using a joint probability or conditional probability respectively [46] [81]. K-Nearest-Neighbor (KNN) is the most basic and popular discriminative technique. The core of KNN depends mainly on measuring the distance or similarity between the tested samples and the training samples. Euclidean distance is the most widely used distance metric in KNN methods. However, several similarity measure techniques can be applied. These include Mahalanobis, Manhattan, Minkowski, Chebychev, Cosine, Correlation, Hamming, Jaccard, etc. [79]. The KNN algorithm determines the k closest matches in the signal space to the target. Then, the

location of the target can be estimated by the average of the coordinates of the k neighbors [58]. Generative localization methods apply statistical approaches, e.g., Hidden Markov Model [41], Bayesian Inference [11], Gaussian Processes [37], on the Wi-Fi fingerprint database. Thus, accuracy can obviously be improved by adding training data. In [37] for instance, Gaussian Processes are used to estimate the signal propagation model through an indoor environment. Generally speaking, fingerprinting-based localization methods can achieve good accuracy. However, the off-line phase demands high effort and is time consuming, which makes it non-practical for large-scale deployments.

RADAR [7] was the first work that utilized a Wi-Fi discriminative fingerprinting method. In RADAR, the interest area is divided into a grid of 1x1 m. RSS measurements are taken at each cell intersection to create the radio map database. Then in the on-line phase, RSS received is compared to the radio map database to estimate the target location. However, it is very time consuming to build up a radio map.

In order to improve the accuracy of fingerprint-based localization, authors of [47] proposed to fuse step counter measurement with location estimation to reduce the calibration efforts. In [48], authors proposed a graph-based, low-complexity sensor fusion approach for ubiquitous pedestrian indoor positioning using mobile devices. However, the system is not robust again positioning failures.

There is a limited number of works that have focused on reducing off-line efforts in learning-based approaches for indoor localization [24] [68] [67]. These approaches reduce the off-line effort by reducing either the number of samples collected at each survey point or the number of survey points or both of them (i.e., reducing the number of collected samples and number of survey points). Then, a generative model is applied to reinforce the sample collection data. In [24] for instance, a linear interpolation method is used. In [68], a Bayesian model is applied. In [67], authors propose a propagation method to generate data from collected samples. In [81], authors combine characteristics of generative and discriminative models in a hybrid model. Although these hybrid models can reduce offline efforts, these models still rely on a number of samples collected from fixed survey points (i.e., labeled samples). To maintain high accuracy, the number of survey points shall be increased in larger environments.

Collecting samples from numerous survey points (i.e., point locations where the ground truth position is known) will become a demanding process, which makes the system unsuitable to large environments. In [69], authors validated the performance

of different individual machine learning approaches for indoor positioning systems. However, they rather compare the results without any deep analysis of the performance difference. Moreover, they did not discuss how ensemble learning approaches could be used to enhance the system performance.

Despite generative approaches can handle the missing value problem, discriminative approaches often achieve better performance. Generally speaking, all the machine learning-based localization methods using fingerprinting can achieve good accuracy, if a large number of labeled samples are available. However, such a sample collection process could take several hours or days for small or big areas, which is very labor expensive and time consuming. Therefore, it is essential to reduce the efforts in offline sample collection procedures while still maintaining high localization accuracy. Our localization approaches incorporate room level prediction results in the localization process to accurately converge to the actual position. Since our approach requires only room level detection (i.e., room recognition), the fingerprint database is built by collecting fingerprint measurements by simply walking randomly through the environment. Thus, we reduce cost, manpower, effort, complexity and survey time in building the learning phase compared to traditional fingerprinting-based and landmark-based localization approaches.

In addition to radio signals, the earth's magnetic field MF_{geo} and magnetic field fluctuations in indoor environments can also be potentially leveraged for indoor localization. Several works show the feasibility of using anomalies of the magnetic field to provide indoor localization [114] [98] [53] [97]. However, due to ferromagnetic material and electrical objects, magnetic signatures have many ambiguities in indoor environments (i.e., similar magnetic field value in different locations). Therefore, MF measurements should be fused with other indoor sensor measurements to derive accurate locations. In this thesis, we provide a probabilistic approach to integrate Wi-Fi, MF readings and information about transitions between rooms to achieve room level localization.

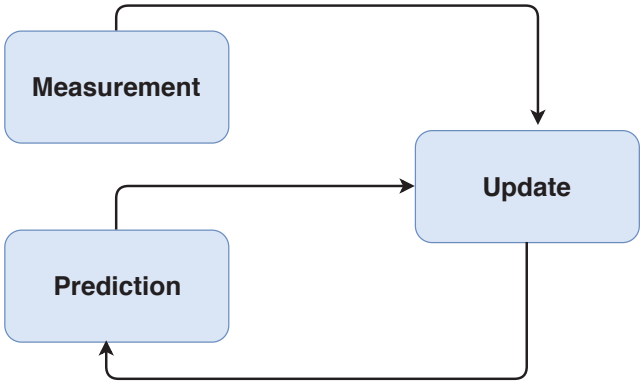


Figure 2.6: Kalman Filter.

2.5 Mathematic Models for Indoor Localization

2.5.1 Kalman Filters

Kalman filter is a popular and widely used recursive algorithm for data fusion or filtering. Kalman filter (KF) was published by Kálmán in 1960 [57]. Abstractly, Kalman filtering can be viewed as a method for fusing noise estimates of some unknown value to obtain a more precise estimate of that value. KF is a filtering algorithm to estimate the state of a linear system, such as the position and velocity of a vehicle. Figure 2.6 shows the three phases of KF: Prediction, measurement, and update. In the prediction phase, the current state is inferred according to the system's state model and based on the previous system state. To verify this prediction, a measurement of the system state is performed. The new system state is then updated by a linear combination of the measurement and the prediction, weighted by further information relating to the process and measurement error.

The KF addresses the general problem of trying to estimate the system state X_k of a controlled process that is governed by a linear dynamic stochastic equation. We consider indoor tracking as a filtering problem, in which the position of a mobile device can be predicted from a stream of noisy environmental measurements. Therefore, the indoor tracking problem has two main elements:

- a system state vector $X_k \in R^m$, which includes the two dimensional coordinates of the mobile device at time k. Vector X_k is updated by the system equation as follows:

$$X_k = F(X_{k-1}) + \mu_{k-1}, \tag{2.6}$$

where the variable μ represents the process noise. Function F describes the transition of the system from the state at time $t = k-1$ to the current state at time $t = k$.

- a discrete set $Z_{1:k} \in R^n$ of environmental measurements from $t = 0$ to time $t = k$. Set Z_k can be expressed as follows:

$$Z_k = H(X_k) + v_k, \tag{2.7}$$

where the variable v represents the measurement noise. Function H describes the relation between the state X_t to the measurement Z at time $t = k$. Variables μ and v are assumed to be independent of each other and normally distributed.

From the Bayesian perspective, KF estimates $p(X_k \mid Z_{1:k})$ which is the probability of X_k being the current system state given that the system perceives a set of observation $Z_{1:k}$. Thus, $p(X_k \mid Z_{1:k})$ is calculated by considering the current system state X_k, the set of available measurements $Z_{1:k}$, and an initial Probability Density Function PDF over the state space of the system $p(X_0)$ [63]. From the Bayes' rule, $p(X_k \mid Z_{1:k})$ can be calculated as follows:

$$p(X_k \mid Z_{1:k}) = \frac{p(Z_k \mid X_k) \cdot p(X_k \mid x_{k-1})}{p(X_k \mid Z_{1:k-1})} \tag{2.8}$$

Equation 2.8 shows that the current state X_k depends on the previous state X_{k-1}. Therefore, the state prediction probability $p(X_k \mid X_{k-1})$ can be computed from the dynamic Equation 2.6. Whereas, the measurements probability $p(Z_k \mid X_k)$ can be computed from the measurement Equation 2.7. The PDF estimation showed in Equation 2.8 is a general conceptual solution, which can not be derived analytically. However, by considering certain assumptions, some algorithms still can achieve possible optimal and sub-optimal solutions such as Linear Kalman Filter, Extended Kalman Filter, and Particle Filter.

Linear Kalman Filter (LKF)

LKF assumes that system equation 2.6 and measurement observation equation 2.7 are linear and all the probability distributions, including noise in system and measurement observation equation are Gaussian distributed.

Process noise μ and measurement observation noise υ are assumed to be independent and Gaussian distributed with zero mean as follows:

$$p(\mu) \sim \mathcal{N}(0, Q) \tag{2.9}$$

$$p(\upsilon) \sim \mathcal{N}(0, R) \tag{2.10}$$

where Q and R are the noise covariance matrix.

In LKF, the system current state X_k is computed based on two steps as follows:

- The first step computes the state prediction by following two equations:

$$\hat{X}_k = F\hat{X}_{k-1} + \mu_{k-1}, \tag{2.11}$$

$$P_k = FP_{k-1}F^T + Q. \tag{2.12}$$

  where Equation 2.11 computes the current system state, and Equation 2.12 projects the covariance estimate forward.

- The second step computes the following equations:

$$K_k = P_k H^T (HP_k H^T + R)^{-1}, \tag{2.13}$$

$$\hat{X}_k = \hat{X}_k + K_k(Z_k - H\hat{X}_k), \tag{2.14}$$

$$P_k = (I - K_k H)P_k. \tag{2.15}$$

  where Equation 2.13 computes the discrepancy between the actual measurement and the predicted system state. This value is called Kalman Gain. Equation 2.14 adjusts the estimation of the current state based on Kalman Gain, the predicted state, and system equation. Equation 2.15 updates the state error covariance.

Extended Kalman Filter (EKF)

EKF assumes that system equation 2.6 and measurement observation equation 2.7 are non-linear. However, all the probability distributions, including noise in system and measurement observation equation are assumed Gaussian distributed. The nonlinear system equation and observation equations are linearized as follows:

$$F(X_{k-1}) \approx F(X'_{k-1}) + J_F(X'_{k-1})(X_{k-1} - X'_{k-1}) \tag{2.16}$$

$$H(X_k) \approx H(X'_k) + J_H(X'_k)(X_k - X'_k) \tag{2.17}$$

where J_F and J_H are Jacobian of F and H respectively. J_F and J_H can be computed as follows:

$$J_F = \begin{pmatrix} \frac{\partial F_1}{\partial X_1} & \frac{\partial F_1}{\partial X_2} & \cdots & \frac{\partial F_1}{\partial X_n} \\ \frac{\partial F_2}{\partial X_1} & \frac{\partial F_2}{\partial X_2} & \cdots & \frac{\partial F_2}{\partial X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial X_1} & \frac{\partial F_n}{\partial X_2} & \cdots & \frac{\partial F_n}{\partial X_n} \end{pmatrix},$$

$$J_H = \begin{pmatrix} \frac{\partial H_1}{\partial X_1} & \frac{\partial H_1}{\partial X_2} & \cdots & \frac{\partial H_1}{\partial X_n} \\ \frac{\partial H_2}{\partial X_1} & \frac{\partial H_2}{\partial X_2} & \cdots & \frac{\partial H_2}{\partial X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial H_n}{\partial X_1} & \frac{\partial H_n}{\partial X_2} & \cdots & \frac{\partial H_n}{\partial X_n} \end{pmatrix},$$

After computing J_H and J_F, EKF adopts a two step procedure to estimate the state of the system:

- The first step computes the system state prediction by the following equations:

$$\hat{X}_k = F(\hat{X}_{k-1}) \tag{2.18}$$

$$P_k = J_F(X_{k-1})P_{k-1} + Q. \tag{2.19}$$

  where Equation 2.18 computes the current system state. Equation 2.19 projects the covariance estimate forward.

- The second step computes the following equations:

$$K_k = P_k J_H^T (J_H(\hat{X}_k) P_k J_H^T(\hat{X}_k) + R)^{-1}, \tag{2.20}$$

$$\hat{X}_k = \hat{X}_k + K_k(Z_k - H\hat{X}_k), \tag{2.21}$$

$$P_k = (I - K_k J_H) P_k. \tag{2.22}$$

  where Equation 2.20 computes the Kalman Gain. Equation 2.21 adjusts the estimation of the current state based on Kalman Gain, the predicted state, and system equation. Equation 2.22 updates the state error covariance.

2.5.2 Particle Filters

Particle filters and Sequential Monte Carlo (SMC) methods allow for Bayesian inference in complex dynamic state-space models. Particle filters are commonly applied to solve filtering problems. In filtering problems, the objective is to estimate the latent states of a stochastic process given a sequence of observations. The particle filter was introduced as a numerical approximation to solve nonlinear filtering problems. Figure 2.7 presents an illustration of the particle filter. The nonlinear filtering problem is to make approximations of the hidden state of the system from a set observations. System state approximations are inferred by computing the posterior distribution (PDF) for the state vector given the set of observations at that time. Particle filters operate on a set of randomly sampled values which are referred as particles. In each iteration of the particle filter, particles are propagated over time to track the PDF of the state (i.e., predicted particles). Then, each particle is evaluated (i.e., likelihood measurement) and each is assigned a weight in relation to its posterior probability (i.e., weighting). To improve its accuracy, SMC techniques resample useful particles according to their weights (i.e., resampling). To deal with non-linear state-space models and non-Gaussian distributions, particle filters are adopted by using Monte Carlo simulations to represent the required PDF.

The dynamic estimation problem assumes two underlying models: the dynamic state model and the measurement model.

The dynamic state model represents the state vector over time. It can be expressed as follows:

$$x_t = f_{t-1}(x_{t-1}) + v_{t-1}, \tag{2.23}$$

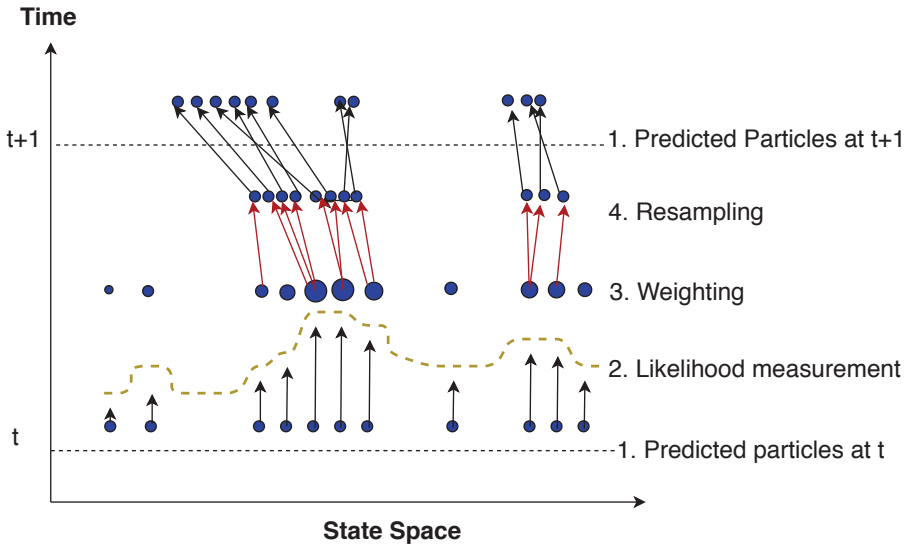


Figure 2.7: Illustration of the Particle Filter

where x_t represents the state vector of the system. Variable t describes the iteration over time and f_{t-1} is a known non-linear function. Variable v_{t-1} is the noise sequence introduced by the process.

The measurement model describes the measurements observed over time in each state of the system. It can be written as follows:

$$z_t = h_t(x_t) + w_t, \tag{2.24}$$

where z_t is the measurement vector observed at time step t. Function h_t is a known non-linear function. Variable w_t is a white noise sequence describing measurements noise.

Particle filter approaches attempt to build the posterior probability distribution function (PDF) of the state vector considering all the observed information. The posterior PDF at time t can be expressed as $p(x_t \mid Z_t)$, where Z_t indicates all the measurements observed so far. Bayesian recursive filters such as particle filters consists of two phases: the prediction phase and the update phase.

In the prediction phase, the posterior PDF is propagated from time $t-1$ forwards to time t. Thus, if $p(x_{t-1} \mid Z_{t-1})$ is available, then the prior PDF of the state vector at time

t can be obtained by the Chapman-Kolmogorov equation [30] as follows:

$$p(x_t \mid Z_{t-1}) = \int p(x_t \mid x_{t-1}) \cdot p(x_{t-1} \mid Z_{t-1}) dx_{t-1}. \tag{2.25}$$

The prior PDF can be updated to incorporate the observed measurements at time t. Thus, the posterior PDF of the state vector can be inferred by the general Bayesian update recursion as follows:

$$p(x_t \mid Z_t) = \frac{p(z_t \mid x_t) \cdot p(x_t \mid Z_{t-1})}{p(z_t \mid Z_{t-1})}, \tag{2.26}$$

where:

$$p(z_t \mid Z_t) = \int p(z_t \mid x_t) \cdot p(x_t \mid Z_{t-1}) dx_t, \tag{2.27}$$

$$p(z_t \mid z_{1:t-1}) = \int_{\mathbb{R}^{nx}} p(z_t \mid x_t) \cdot p(x_t \mid z_{1:t-1}) dx_t, \tag{2.28}$$

$$p(x_{t+1} \mid z_{1:t}) = \int_{\mathbb{R}^{nx}} p(x_{t+1} \mid x_t) \cdot p(x_t \mid z_{1:t}) dx_t, \tag{2.29}$$

where Equation 2.26 follows the Bayes' law, and Equations 2.28 and 2.29 follow the law of total probability.

Importance Sampling

Importance Sampling (IS) is a Monte Carlo integration method to estimate a density or distribution from a set of samples. IS is used to solve high-dimensional integration problems when analytical solutions are unobtainable. IS is often used to approximate expected values of random variables.

Considering a set of N random samples $\{x_{t-1}^i\}_{i=1}^N$ from the posterior PDF $p(x_{t-1} \mid Z_{t-1})$ is available.

The prediction phase consists propagating $\{x_{t-1}^i\}_{i=1}^N$ from time $t-1$ through the system state model defined in Equation 2.23. Thus, the new set of samples (i.e., prior samples) can be written as follows:

$$x_t^i = f_{t-1}(x_{t-1}^i, v_{t-1}^i), \tag{2.30}$$

where v_{t-1}^i represents the system noise.

In time t, the prior samples are updated based on the observations z_k. Thus, an associated weight w_t^i is computed for each sample x_{t-1}^i. The associated weights' w_t^i likelihood is evaluated at the value of the prior sample as follows:

$$w_t^i = p(z_t \mid x_t^i), \tag{2.31}$$

The sample set $\{x_{t-1}^i\}_{i=1}^N$ can be written as a Monte Carlo density approximation as follows:

$$p(x_t \mid Z_{t-1}) \approx \frac{1}{N} \cdot \sum_{i=1}^{N} \delta(x_t - x_t^i), \tag{2.32}$$

and the posterior weighted PDF can be represented as follows:

$$\begin{aligned} p(x_t \mid Z_t) &\approx \sum_{i=1}^{N} w_t^i \cdot \delta(x_t - x_t^i), \\ p(x_t \mid Z_t) &\approx \frac{1}{N} \cdot \sum_{i=1}^{N} \delta(x_t - x_t^i). \end{aligned} \tag{2.33}$$

Substituting Equation 2.26 into Monte Carlo density approximation described in Equation 2.32, we have:

$$\begin{aligned} p(x_t \mid Z_t) &= \frac{p(z_t \mid x_t) \cdot p(x_t \mid Z_{t-1})}{p(z_t \mid Z_{t-1})}, \\ p(x_t \mid Z_t) &\approx p(z_t \mid x_t) \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{\delta(x_t - x_t^i)}{p(z_t \mid Z_{t-1})} \\ p(x_t \mid Z_t) &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(z_t \mid x_t^i) \cdot \delta(x_t - x_t^i)}{p(z_t \mid Z_{t-1})} \\ p(x_t \mid Z_t) &\approx \sum_{i=1}^{N} w_t^i \cdot \delta(x_t - x_t^i), \end{aligned} \tag{2.34}$$

where N is the number of particles. x_t^i is the ith particle and w_t^i is its associated weight at time t. The associated weights can be recursively calculated as follows:

$$w_t^i \propto w_{t-1}^i \cdot p(z_t \mid x_t^i), \tag{2.35}$$

where $p(z_t \mid x_t^i)$ is the $likelihood$ function calculated from the observation measurements vector at time t.

Resampling Process

The resampling process aims to prevent the degeneracy of the propagated particles from a set of particles P_t by generating a new set of particles $\hat{P}_t$. Thus, the resampling process modifies the weighted approximate density p to an unweighted density $\hat{p}$ by eliminating particles with low importance weights (i.e., small associated weight) by multiplying particles having high importance weights (i.e., high associated weight). The new density $\hat{p}$ is called the proposal distribution. Therefore, $p(X_t \mid q_{1:t}) = \sum_{i=0}^{N_s} w_t^i \delta(X_t - X_t^i)$ is replaced by $p(\hat{X}_t \mid q_{1:t}) = \sum_{i=0}^{N_s} \frac{n_i}{N_s} \delta(\hat{X}_t - \hat{X}_t^i)$, where $\hat{X}_t$ is the state vector computed from the new set of particles $\hat{X}_t^i$ and n_i is the number of copies of particle $\hat{X}_t^i$ from $\hat{P}_t$. There are many methods to generate $\hat{P}_t$ [99]. One of the most widely used and efficiently implementable is the systematic resampling method [56]. Our proposed tracking algorithms implement the resampling process based on the systematic method.

The systematic resampling method prevents the degeneracy of the propagated particles by modifying the set P_t to $\hat{P}_t$. Particles from P_t with higher weights are more likely to be included in the new set of particles $\hat{P}_t$. Thus, in the next iteration, more particles will be propagated in zones with large probability masses [56]. Before resampling, the weights W_t^k are normalized, i.e., $\sum_{k=1}^{N_p} W_t^k = 1$. Then, a set of N_p numbers u_t^n is generated from an uniform distribution. This set of numbers is used to select N_p particles from P_t. Thus, the particle x_t^n is selected in the n-th iteration if the following condition is satisfied:

$$S_t^{m-1} < u_t^n \leq S_t^m, m = 1, ..., N_p, \tag{2.36}$$

where

$$S_t^m = \sum_{k=1}^{m} W_t^k, \tag{2.37}$$

The interval $(0,1]$ is divided into N_p disjoint sub-intervals $(0, 1/N_p] \cup ... \cup (1 - 1/N_p, 1]$. Then, u_t^1 is generated as a random number from the uniform distribution on $(0, 1/N_p]$. The remaining u_t^n numbers are obtained from u_t^1 as follows:

$$\begin{aligned} u_t^1 &\sim U(0, 1/N_p], \\ u_t^n &= u_t^1 + \frac{n-1}{N_p}, \quad n = 2, 3, ..., N_p, \end{aligned} \tag{2.38}$$

After generating the set of u_t^n numbers, the new set of particles $\hat{P}_t$ is generated by

selecting N_p particles from P_t based on the condition presented in Equation 5.4. Resampling is a fundamental process for particle filters. Without resampling, particle filters will produce a degenerate set of propagated particles (i.e., most of the particles with negligible weight).

Application of Particle Filters in Indoor localization and Tracking

In indoor localization and indoor tracking applications, the system state can be represented as a state vector containing position, velocity, acceleration, etc. of a moving target. The set of observations can be obtained from either embedded sensors (e.g., inertial sensors), or from external sensors (e.g., measuring ranges to anchor nodes).

The objective is to determine the posterior distributions of the system's states given some noisy observations. The posterior probability is expressed as a set of weighted particles. Thus, the posterior probability distribution is computed based on some observation O_t at time t [20]. At time t, the particle system state vector X_t can be written as:

$$X_t = [x_t, y_t, z_{l_t}, \ell_t, \theta_t], \tag{2.39}$$

where (x_t, y_t) defines the 2-dimensional position of the target object, z_{l_t} is the zone where the target is located, θ_t is the heading orientation, and ℓ_t is the displacement length. Our tracking algorithms proposed in this thesis are based on the system state model defined in Equation 2.39.

At time t, the set of particles can be expressed as:

$$P_t = [X_t^i, W_t^i], i = 1, ..., N, \tag{2.40}$$

where N is the number of particles, X_t^i is the state vector, and W_t^i is the associated weight of the i-th particle at time t. Based on Equation 2.33, the posterior probability given a sequence of observations $p(X_t \mid O_{1:t})$ can be defined as:

$$p(X_t \mid O_{1:t}) \approx \sum_{i=0}^{N} w_t^i \delta(X_t - X_t^i), \tag{2.41}$$

where x_t^i is the i-th particle, and w_t^i is the associated weight at time t. Therefore,

based on Equation 2.34, the associated weights w_t^i can be computed as follows:

$$w_t^i \propto w_{t-1}^i * p(O_t \mid X_t^i), \tag{2.42}$$

where $p(O_t \mid X_t^i)$ is the *likelihood* function calculated from the observation vector O_t at time t. Therefore, the particle filter prediction function can be written as:

$$X_t = G \cdot X_{t-1} + \eta, \tag{2.43}$$

where

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \eta = \begin{pmatrix} \ell_t \cdot cos(\theta_t) \\ \ell_t \cdot sin(\theta_t) \\ z_{l_t} \\ \ell_t \\ \theta_t \end{pmatrix}$$

State vector X_t^i of each particle is updated from the particles at the previous time interval X_{t-1}^i based on Equation (2.43).

The integration of data and knowledge from several sources is known as data fusion. Particle filter is used as data fusion techniques to combine data from multiple sensors and related information from associated databases to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone [44]. Besides the methods solely relying on PDR and radio-based positioning, some work has been proposed to combine information from multiple localization methods. In [77], authors proposed a fingerprinting-based solution by combining digital compass and Wi-Fi information. The authors of [50] proposed a tracking system by exploiting particle filter features. Incorporating environmental knowledge can further improve the positioning and tracking accuracy. Typically, standard particle filter approaches operate on a continuous two-dimensional map. In [48], authors propose a graph-based method to model the environment. In this approach, the commonly used two position coordinates is reduced to a single node identifier. The floor plan graph-based model is able to remove degrees of freedom from the map wherever they are irrelevant to the localization task. The graph-based model is used to sample only valid movements. This way, a lower number of particles is used to represent pedestrian motion. Despite these approaches are able to reduce the system

state representation algorithmic complexity, building the graph-based model requires high effort and it is time consuming, which makes it non-practical for large-scale deployments. To address this problem, in this thesis we exploit machine learning for landmark detection with zone level accuracy.

Data fusion of wireless radio-based approach and the PDR-based approach are two mainstream easy-to-use solutions, which could normally generate meter-level accuracy. However, recently some new efforts consider computer vision-based approaches to assist smartphone indoor localization were developed. Recent advances in computer vision techniques and the ubiquity of smartphone cameras hold promise for improving localization accuracy to sub-meter level. In [115], authors have proposed an sensor-enriched, image-based indoor localization system, which leverages minimal sensor-enriched photos to map the image-generated Place of Interest (POI) model into the physical space. They combine Wi-Fi reading, IMUs measurements, floor plan, and camera images to generate accurate localization results in sub-meter level. In [43], authors have designed an indoor localization system for performing fine localization and less latency with more priori information, including tile angle and the relative height between camera optical center and origin in reference coordinate system. The proposal was claimed to keep the error in sub-meter range and get the fine direction angle of the device. Despite authors of [43] and [115] claim to achieve high accuracy, the target device requires sophisticated hardware and software such as image processing algorithms, high definition cameras.

In [107] authors used radio propagation models to reduce the efforts during the calibration process. The probable positions are inferred by using discrete probability distributions. Afterwards, the position is computed from the set of most probable estimated positions. In [66], the authors propose to fuse wireless signal measurements with IMU readings. Then, the position is determined by computing based on the most probable wireless signals measurement and the pedestrian motion pattern at that position. Although authors claim high accuracy, the transition probability definition method remains unclear. Additionally, the applicability of the approach relies on the fidelity of the PDR method.

Particle filters are normally known as Sequential Monte Carlo methods [32], which were originally designed to solve statistic problems. Particle filters are able to approximate any probability density function, which can be regarded as a sequential analogue of Markov chain Monte Carlo (MCMC) methods. Although particle filter

is a statistic approach, it has been successfully applied in many applications, such as Monte Carlo localization of mobile robots [40], simultaneous localization and mapping (SLAM) [73], etc. However, the benefits of exploring particle filters into the reinforcement learning domain seem to be missing so far. Thus, in this thesis, we focus on particle filter techniques to integrate data and knowledge from several sources (e.g., room prediction results, radio-based ranges, IMUs, and coarse-grained floor plan information) to achieve accurate and reliable indoor localization.

2.5.3 Supervised Learning

The supervised learning process is done by using prior knowledge of the output values of the samples. Thus, supervised learning methods learn a function that given some samples of data and desired outputs, best approximates the relationship between input and output in new observable data. Unsupervised learning does not have knowledge of the desirable outputs. Unsupervised learning methods infer the natural structure present within a set of data samples. The reinforcement learning process is done by capturing the most important aspects of the problem while an agent is interacting with the environment. In this thesis, we focus on supervised and reinforcement learning methods. In the following paragraphs, we shortly describe the machine learning algorithms that are used in this work to perform the localization tasks.

Decision Tree

A decision tree is a hierarchical structure for classifying objects, it is composed of nodes that correspond to primitive classification decisions. Decision trees are non-parametric supervised learning method used for classification and regression. Decision Tree creates the classification model by learning simple decision rules inferred from the data features. Some strengths of decision tree models are as follows:

- Classification rules are simple to interpret. Moreover, classification rules can be visualized.
- Data input does not require complex methods such as normalization before feeding the classification model.
- A decision tree method is able to handle numerical and categorical data.

Decision tree models create a flowchart-like structure based on the attribute values. This flowchart-like structure supports the decision rules for classification and regression. Figure 2.8 shows an example of a decision tree classifier structure. In decision trees, the internal nodes represent features (i.e., attributes), whereas branches represent the decision rules. Each leaf node represents an outcome. The root node is the topmost node in the decision tree structure. Further details about decision trees classifiers can be found in [12].

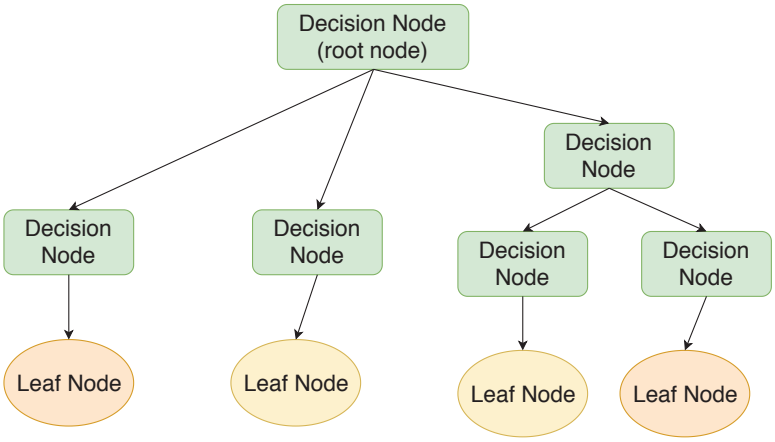


Figure 2.8: Decision Tree Learner Structure

Naive Bayes (NB)

Naive Bayes-based classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Bayes theorem states that given class variable y and dependent feature vector x_1 through x_t:

$$p(y \mid x_1, ..., x_t) = \frac{p(y) \cdot p(x_1, ..., x_t \mid y)}{p(x_1, ..., x_t)}, \tag{2.44}$$

where $p(y \mid x_1, ..., x_t)$ represents the class probability y given that the feature vector $(x_1, ..., x_t)$ is observed. $p(y)$ is the probability of class y, and $p(x_1, ..., x_t)$ is the probability of observing the feature vector $(x_1, ..., x_t)$.

Considering the naive conditional assumption $p(x_i \mid y, x_1, ..., x_t) = p(x_i \mid y)$, Equation

2.44 is simplified as follows:

$$p(y \mid x_1, ..., x_t) = \frac{p(y)\prod_i^t p(x_i \mid y)}{p(x_1, ..., x_t)}, \tag{2.45}$$

since $p(x_1, ..., x_t)$, the classification rule is as follows:

$$p(y \mid x_1, ..., x_t) \propto p(y)\prod_i^t p(x_I \mid y) \tag{2.46}$$

Further details about Naive Bayes classifiers can be found in [11]

K-Nearest Neighbors (KNN)

KNN is a non-parametric method used for classification and regression. Non-parametric means that predictions and regressions are made without explicit assumptions about the functional form of hypothesis. This avoids wrong assumptions about the underlying distribution of the data. In classification problems, the input of KNN consists of the k closest training samples in the feature space. Classification is estimated by considering a simple majority vote of the most similar nearest neighbors classes. The similarity is defined according to a distance metric between two data points. A common choice is the Euclidean distance as follows:

$$d(x, x') = \sqrt{(x_1 - x')^2 + ... + (x_n - x')^2}, \tag{2.47}$$

where $d(x, x')$ represents the Euclidean distances between data point x and x'. Despite Euclidean distance is a common choice, other similarity measures including Manhattan, Chebyshev and Hamming distance can be applied to different classification problems.

The optimal choice of the K value depends on the data and the classification problem. In general, a large value of K tackles the effects of noise. However, a large value of K can detrimentally affect the classification accuracy. Further details about KNN classification model can be found in [79]

Support Vector Machine (SVM)

SVM is a supervised learning model with associated learning algorithms that analyze data used for classification and regression. However, it is mostly used in classification problems. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new data measurements to one category or the other, making it a non-probabilistic binary linear classifier. The goal of the SVM algorithm is to create hyperplanes in an n-dimensional space (n is the number of features). Hyperplanes are created to separate data points based on their similarities. Thus, hyperplanes are decision boundaries that support the classification process. Data points falling on either side of the hyperplane are attributed different classes. The dimension of the hyperplane depends upon the number of features. Thus SVM represents each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. Then, SVM performs classification by finding the hyper-plane that better differentiates the classes.

Support vectors are data points close to the hyperplanes. Since support vectors influence the position and orientation of the hyperplanes, support vectors are used to maximize the margin of the data points in the classifier. SVM classifier achieves high performance in high dimensional spaces (i.e., effective in cases where number of features is high). Further details about SVM classifier can be found in [72].

Artificial Neural Network (ANN)

Artificial Neural Networks is a class of feed-forward artificial neural network. A MLP consists of at least three layers of nodes. Figure 2.9 shows a graphical representation of a MLP learner with a single hidden layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP is a supervised learning model that learns a function $f(\cdot) : R^m \rightarrow R^o$, where m is the number of input dimensions (i.e., features) and o is the number of dimensions for output (i.e., classes). MLP is able to learn a non-linear function approximator to solve either classification or regression problems. Further details about MLP can be found in [90]

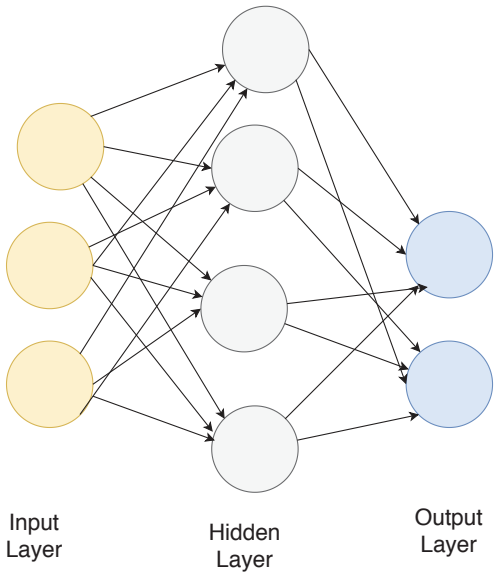


Figure 2.9: Single hidden layer MLP graphical representation.

Soft Voting (SV)

SV is one of the simplest ensemble predictors. It combines predictions from multiple individual machine learning algorithms. It works by first creating two or more standalone prediction models from the training dataset. A SV classifier can then be used to wrap the models and average the predictions of the sub-models when asked to make predictions for new data. The key idea of SV is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to make a prediction. Further details about the SV model can be found in [92]

2.5.4 Reinforcement Learning

The key idea of reinforcement learning (RL) is to capture the most important aspects of the real problem which is facing a learning agent (LA) interacting with its environment. Such LA must be capable to perceive the state of the environment and must be able to take actions that affect those states. The LA has to achieve goals related to the state of the environment. Reinforcement learning involves learning the combinations situation-action which lead to maximize a numerical reward. In RL, a learner agent (LA) is not told which actions to perform. In RL, the LA must discover which actions yield the most reward by trying them out. Actions influence later inputs of the system.

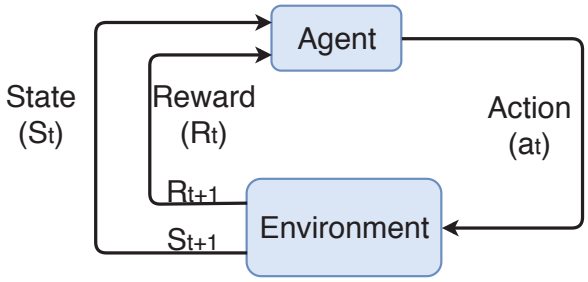


Figure 2.10: Interaction between the Agent and the Environment in RL.

Actions can affect not only immediate reward but also future situations and rewards. Thus, RL problems are defined by these characteristics [84]:

- Future inputs of the system are influenced by current actions.
- The reinforcement learning system does not have any direct instruction about actions to perform.
- The consequences about actions are unknown.

The key idea in RL is to capture the important aspects of the real problem by making a LA interacting with the environment to achieve a goal [84]. The LA must be able to sense the state of the environment and must be able to perform actions regarding this state.

Elements of Reinforcement Learning

In addition to the LA and the environment, there are four elements in a RL system: Policies, reward signal, value function, and model of the environment.

- Policies define the behaviour of the LA at a given time step. Thus, a policy is a mapping between environment states and actions to take in those states. Policies determine the behavior of the LA. In some RL systems, policies can be modeled as a lookup table, however, in other systems, determining the policy can involve extensive computation processes such as deep learning.
- Reward signals define the goal in a RL problem c. At each time step, the LA receives a numerical reward from the environment. This numerical reward is

called the reward signal. The objective of the LA is to maximize the total reward received over the operation of the system. The reward signal is sent to the LA at each time step. Its value depends on the LA's action and the current state of the environment. Therefore, reward signals determine how profitable is to perform an action in each state of the environment.

- The value function specifies how profitable is an action in the long term time. Thus, the value function calculates the total amount of reward that the LA can accumulate over the future. Thus, the value function indicates the long-term desirability of states by considering the states that are likely to follow, and the associated rewards in those states.
- The model of the environment mimics the behavior of the environment. It enables inferences about the behaviour of the environment.

Q-Learn Reinforcement Learning Method

Q-learning is a method that allows agents to learn how to act optimally in controlled Markovian domains [109]. Thus, Q-learning provides LA with the capability of learning the suitable behaviour in Markovian domains by experiencing the consequences of actions.

Consider a LA moving along a discrete, finite system state environment. The LA must choose one from a discrete set of actions at every time step. At time step n, the LA sense the state $x_n \in X$ of environment. Then, the LA selects an action $a_n \in A$ according to the current environment state. The LA receives a probabilistic reward r_n. The value of $r_n = R_{x_n}(a_n)$ depends on the action and the state of the environment. The state of the environment changes probabilistically to y_n according to the law:

$$p(y_n = y \mid x_n, a_n) = p_{x_n y}(a_n), \tag{2.48}$$

The LA must determine the optimal policy that maximizes the total expected reward. Thus, under a policy π, the value of the state x can be calculated as follows:

$$V^{\pi}(x) = R_x(\pi(x)) + \gamma \sum_{y} p_{xy}(\pi(x)) V^{\pi}(y), \tag{2.49}$$

where γ is a reward discounted factor ($0 < \gamma < 1$). The LA receives a reward $R_x(\pi(x))$ for executing action $\pi(x)$. Then, the environment state changes to $V^{\pi}(y)$ with probability

$p_{xy}(\pi(x))$. Therefore, according to the Dynamic Programming Theory (DP) [9], there is at least one optimal policy π^* which is such that:

$$V^*(x) = V^{\pi^*}(x) = \max_a \left\{ R_x(a) + \gamma \sum_y p_{xy}(a) V^{\pi^*}(y) \right\} \tag{2.50}$$

DP proposes several methods to compute V^* and π^* from known $R_x(a)$ and $p_{xy}(a)$ values. However, the Q-learning approach provides an approach to determine π^* without knowing $R_x(a)$ and $p_{xy}(a)$. The Q-learning method determines step-by-step the optimal policy as follows:

$$Q^\pi(x,a) = R_x(a) + \gamma \sum_y p_{xy}(\pi(x) V^\pi(y)), \tag{2.51}$$

where $Q^\pi(x,a)$ defines the Q-value for a policy π. Thus, the Q-value is the expected reward for performing action a at state x by following policy π.

In the Q-learning method, the LA experiences a sequence of episodes. In each episode n the LA executes the following processes:

- Observation of the current environment state x_n.
- Selection and performing of action a_n.
- Observation of the subsequent state y_n.
- Reception of the immediate reward r_n.
- Computation of Q_{n-1} value using the learning factor α_n, as follows:

$$Q_n(x,a) = \begin{cases} (1-\alpha_n) Q_{n-1}(x,a) + \alpha \cdot (r_n + \gamma V_{n-1}(y_n)) & \text{if } x = x_n \\ Q_{n-1}(x,a) & \text{otherwise,} \end{cases} \tag{2.52}$$

where $V_{n-1}(y) = \max_b \{Q_{n-1}(y,b)\}$ and the initial Q-values ($Q_0(x,a)$) for all states and actions are known.

2.6 Conclusions

In this Chapter, we provided a detailed description of the concepts that we use during this thesis. Thus, in this Chapter, we reviewed the theoretical background and

work related to indoor localization methods. In Section 2.2, we introduced two wireless communication standards which are widely used for indoor localization (IEEE 802.15.4a and the IEEE 802.11). Additionally, we introduced network architectures. We focused on Multi-Access Edge Computing technology and its reference framework. MEC architecture brings the storage and processing capabilities in close proximity to the user. Thus, network latency, network bandwidth, limited battery and processing resources in mobile devices are handled by MEC-based architectures. We reviewed some key aspects and components of the Internet of Thinks network. In Section 2.3, we described some electromechanical devices used to measure relative movement. These devices are referred as Inertial Measurements Units, which are used in Pedestrian Dead Reckoning systems for indoor localization and tracking. In Section 2.4, we descried signal based localization methods. Because of its ubiquitous availability, indoor ambient signals are often used in indoor positioning. In Section 2.5, we introduced some mathematical models for indoor localization including LKL, EKF, and PF. Additionally, we introduced machine learning algorithms such as supervised and reinforcement learning methods. We focused on the Q-learning algorithm which is used in our work.

Part I

Enhanced Methods for Indoor Localization and Tracking

In this part, we present enhanced methods to provide high localization and tracking performance. Proposed methods are implemented on a terminal-based architecture. The main idea behind terminal-based localization systems is that the localization and tracking methods are implemented on the target terminal devices. Thus, localization and tracking do not require any assistance of an additional entity such as external servers. However, the vision of indoor localization and tracking on commodity mobile devices entails big challenges. For example, the noise and low sampling rate readings in sensors of commodity mobile devices introduce errors in the process of numeric integration during localization [50]. In this part, we also focus on providing robustness to localization failures. Robust indoor localization entails providing methods to prevent and recover the system from localization failure problems. Thus, in this Part, we focus on enhanced solutions for localization and tracking with failure recovery methods. The proposed localization algorithms are designed in a terminal-based system which does not require assistance of any third part server.

3

Enhanced Particle Filters for Range-based Tracking

3.1 Introduction

In this chapter, we present an indoor localization approach to support continuous positioning and tracking. Our approach is able to provide high accuracy by fusing IMUs, radio, and floor plan information in an enhanced particle filter. The localization approach is formulated in a discretized graph-based representation of the indoor environment. We provide an efficient asynchronous continuous resampling method to mitigate errors caused by off-the-shelf IMUs and Wi-Fi sensors embedded in commodity smartphones. Additionally, we propose an efficient filtering approach for improving the IMU measurements, which is able to mitigate errors caused by inaccurate off-the-shelf IMUs and magnetic field disturbances. The algorithms are designed in a terminal-based system, which consists of commercial smartphones and Wi-Fi access points. We evaluate our system in two complex environments along moving paths. Experiment results show that our tracking method can achieve the

average tracking error of 1.01 meters and 90% accuracy of 1.7 meters.

The main contributions of this chapter are summarized as follows.

- We propose an enhanced particle filter to fuse range information estimated from RSSI, IMUs as well as floor plan information for indoor tracking. RSSI, IMUs, and floor plan information are used to correct the associated weights of the propagated particles in the observation phase of the particle filter.
- We incorporate an asynchronous continuous correction phase in the particle filter. The correction phase is able to mitigate the tracking errors caused by unstable RSSI readings and low sampling frequency experimented in Wi-Fi sensors of commodity smartphones.
- We integrate an enhanced PDR method by considering magnetic field and angular velocity measurements to further improve the heading orientation estimation. PDR methods provide information about the heading orientation of the target mobile device, which is obtained by developing an enhanced digital compass based on magnetic field and gyroscope readings.
- We propose an efficient discretized graph-based method to describe the physical environment. Physical environment information such as floor plan is used to define the areas where the pedestrian is allowed to move.

The rest of the Chapter is organized as follows. The proposed enhanced particle filter is presented in Section 3.2. Section 3.3 presents ranging and PDR methods. Implementation of the terminal-based system is presented in Section 3.4. Section 3.5 presents the evaluation results of our approach. Section 3.6 concludes the chapter.

3.2 An Enhanced Particle Filter with Asynchronous Continuous Correction Phase

Figure 3.1 summarizes the structure of our proposed approach and Figure 3.2 depicts our enhanced particle filter structure. We propose an enhanced particle filter approach by fusing PDR, Wi-Fi, and floor plan information. In this approach, an additional asynchronous continuous correction phase is designed in order to further

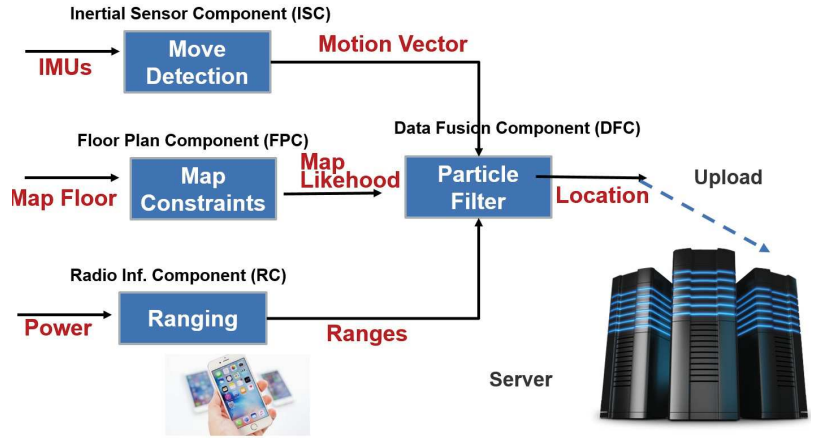


Figure 3.1: Indoor Localization System Architecture.

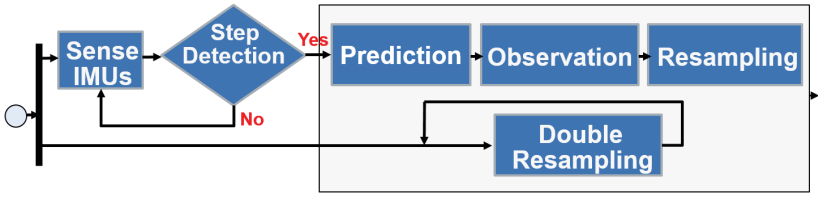


Figure 3.2: Particle Filter Structure

mitigate the errors caused by off-the-shelf Wi-Fi sensors embedded on commodity smartphones. The state vector at time t is defined as follows:

$$X_t = [x_t, y_t, \theta_t, \ell_t], \tag{3.1}$$

where (x_t, y_t) are the Cartesian coordinates of the target object, θ_t is the heading orientation and ℓ_t is the stride length. We define the motion vector as $Mv_t = [\theta_t, \ell_t]$. Thus, the prediction function can be written as

$$X_t = F \cdot X_{t-1} + \eta, \tag{3.2}$$

where

$$F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \eta = \begin{pmatrix} \ell * cos(\theta) & 0 \\ 0 & \ell * sin(\theta) \\ \theta & 0 \\ 0 & \ell \end{pmatrix}$$

$$\theta = \theta^{'} + \varepsilon^{'}$$

$$\ell = \ell^{'} + \varepsilon^{''}$$

Both θ and ℓ values are given by IMUs. Heading orientation and stride length are assumed to interfere by zero-mean Gaussian random noises. Therefore, $\varepsilon^{'}$ and $\varepsilon^{''}$ are the errors introduced in the calculation process of θ and ℓ respectively. State X_t^i vector of each particle is updated based on Equation (3.2) from the particles at the previous time interval X_{t-1}^i. Thus, the new set P_t is calculated from P_{t-1}, and the current system state belief is calculated through four phases as follows:

3.2.1 Prediction Phase

Each particle is updated based on Equation (3.2). Floor plan restrictions are applied in this phase. Any particle is allowed to move through movement-restricted areas, e.g., movement through walls is not allowed. Prediction function (3.2) depends on the motion vector Mv_t.

The heading orientation θ can be statistically described as follows:

$$\theta_t = \hat{\theta}_t + \theta_{bs,t} + \theta_{be,t} + \epsilon_{\theta,t}, \tag{3.3}$$

where $\hat{\theta}_t$ is the actual heading orientation value, $\theta_{bs,t}$ is an angular bias introduced by uncalibrated sensors, $\theta_{be,t}$ is an angular bias due to local magnetic field perturbations, and $\epsilon_{\theta,t}$ is a measured random error [70].

The stride length ℓ can be statistically described as follows,

$$\ell_t = \hat{\ell}_t + \ell_{bs,t} + \epsilon_{\ell,t}, \tag{3.4}$$

where $\hat{\ell}_t$ is the actual stride length value, $\ell_{bs,t}$ is the bias introduced by the use of

uncalibrated sensors, and $\epsilon_{\ell,t}$ is a measured random error [70].

To compensate the bias and error values introduced by the environment and uncalibrated sensors, in this work we assume the heading direction θ and stride length ℓ as random normal variables whose values can be obtained from $\mathcal{N}(\theta_t, \sigma_\theta^2)$ and $\mathcal{N}(\ell_t, \sigma_\ell^2)$, respectively.

3.2.2 Observation Phase

The associated weight w_t^i of the propagated particles must be corrected after updating their positions. The associated weight should be updated based on the likelihood of the observations conditioned on each particle $p(Z_t \mid X_t^i)$ at time t. The observation vector is defined by the estimated ranges to different ANs. Thus, the observation vector at time t is defined as $Z_t = [d_t^j]$, $j = 1...N$, where N is the number of ANs. Then, the probability $p(Z_t \mid X_t^i)$ can be determined as follows:

$$p(Z_t \mid X_t^i) = p(d_t^j \mid X_t^i) \tag{3.5}$$

In this phase, the associated weight w_t^i of each particle is given by the ranging information. The particle at position (x_t, y_t) with low probability to observe d_t^j in their position will be assigned a small weight. In this way, particles with large associated weights will have a stronger contribution in the determination of the state belief of the system. In order to avoid confusion between different likelihoods used in this work, hereafter, we refer to $p(d_t \mid X_t^i)$ as the ranging likelihood and $p(Z_t \mid X_t^i)$ as the overall likelihood.

We can assume that the ranges to different ANs are independent from each other. Therefore, the ranging likelihood can be defined as follows:

$$p(Z_t \mid X_t^i) = \prod_{j=1}^{N} (\hat{d}_{j,t} \mid X_t^i), \tag{3.6}$$

where $\hat{d}_{j,t}$ is the measured distance to the AN j at time t. Hereafter, $p(\hat{d}_{j,t} \mid X_t^i)$ will be referred as the individual likelihood.

Each individual likelihood can be written as:

$$p(\hat{d}_{j,t} \mid X_t^i) = \frac{1}{\sigma_j \sqrt{2\pi}} \exp^{\frac{[\hat{d}_{j,t} - \sqrt{(x^i - x_j)^2 + (y^i - y_j)^2}]^2}{2\sigma_j^2}}, \tag{3.7}$$

where (x_j, y_j) are the coordinates of the jth AN.

To mitigate the influence of ranging errors on the definition of the ranging likelihood $p(d_k \hat{\mid} X_t^i)$, we propose to adopt the same weighting technique used in [62]. The weighting technique magnifies the contribution of the individual likelihood with smaller errors and suppresses the contribution of larger ranging errors. Therefore, the weighted technique is defined on each individual likelihood as follows:

$$p(Z_t \mid X_t^i) = \prod_{j=1}^{N} p(\hat{d}_j \mid X_t^i)^{m_j}, \tag{3.8}$$

where m_j is the exponential weight for the individual likelihood of the jth AN. In general, estimation of larger distances introduces more errors than small distances. Thus, the exponential weight m_j can be defined as inversely proportional to the estimated range outputs [62]:

$$m_j = \frac{\frac{1}{d_j}}{\sum_{n=1}^{N_{ap}} \frac{1}{d_n}}, \tag{3.9}$$

where N_{ap} is the number of ANs.

3.2.3 Resampling Phase

As discussed in Section 2.5.2, the resampling step is a crucial but computationally expensive component of a particle filter approach. After computing the associated weight W_t^i of each particle, we perform a systematic resampling method as introduced in section 2.5.2. The systematic resampling method aims to prevent the degeneracy of the propagated particles by generating a new set of particles. Particles with higher weights are more likely to be included in the new set of particles. Further details about the systematic resampling method can be found in section 2.5.2.

3.2.4 Asynchronous Continuous Correction Phase

Range estimation is often shifted from the ground truth range [61]. Moreover, we mentioned in previous sections that unlike inertial sensors, Wi-Fi sensors can achieve

a sampling frequency of approximately 4Hz. Thus, when particles are propagated in the Prediction phase (i.e., when a step is detected), there is no guarantee to have updated Wi-Fi RSSI information to perform the Observation phase. Additionally, we observe that Wi-Fi RSSI values measured at the smartphone side can fluctuate over time even when the smartphone is held in a static position. Thus, determination of the associated weight of each particle is not accurate since the Observation Phase presented in Section 3.2.2 relays only on ranging information, which is obtained just after performing the Prediction phase. Therefore, individual likelihoods $p(\hat{d}_{j,t} \mid X_t^i)$ are often biased from the real individual likelihood $p(d_{j,t} \mid X_t^i)$.

By performing an asynchronous continuous correction of the associated weight of each particle, we intend to mitigate the errors introduced by Wi-Fi instability and the low sampling rate experimented in smartphones. Thus, the asynchronous continuous correction phase is asynchronously executed whenever new Wi-Fi RSSI information is available. This phase includes an additional systematic resampling method. Thus, this phase continuously evaluates particle weights and perform a resampling process based on the current particle weight values. Hereafter, we refer to the asynchronous continuous correction phase as the double resampling method. The double resampling method implements three main processes as follows:

- The ranging process is executed every time a new Wi-Fi RSSI reading is available.
- The weight of each particle is continuously recalculated and a systematic resampling process is performed based on the current particle weights. In this phase the associated weight is determined based on the individual likelihood ($p(\hat{d}_{j,t} \mid X_t^i)$) and the floor plan information. It is worth to mention that $p(\hat{d}_{j,t} \mid X_t^i)$ is determined by using ranging information.
- Particles are continuously resampled in a systematic method.

To summarize, the double resampling method is aimed to make a continuous correction of the level of influence that each particle contributes to the determination of the state of the system. Algorithm 1 describes the double resampling method.

Algorithm 1 Asynchronous Continuous Resampling

Input : Floor Plan Constraints, $R\hat{S}SI$

Output : Particle's corrected weights

Scan WiFi network.

if *new* $R\hat{S}SI$ *reading is available* **then**

  Determine $R\hat{S}SI$ mean ($mRSSI$) of the latest 4 $R\hat{S}SI$ readings:

  $mRSSI = \frac{\sum_{k=0}^{3} RSS_{t-k}}{4}$

  foreach AN_j **do**

    calculate $d_j = \alpha_j \cdot e^{\beta_j \cdot mRSSI_j}$

  end

  Calculate the individual likelihood: $p(d_j \mid X_t^i)$

  Check position of each particle:

  foreach *Paricle* P_t **do**

    if *Paticle position is in restricted area* **then**

      $W_t = 0$

    end

  end

  Normalize weights of each particle:

  foreach *Paricle* P_t **do**

    $W_t = \hat{W}_t / \sum_{n=1}^{N} \hat{W}_n^i$

  end

  Resample particles P_t based on systematic resampling method.

end

Go to 1

3.3 Ranging and PDR Methods

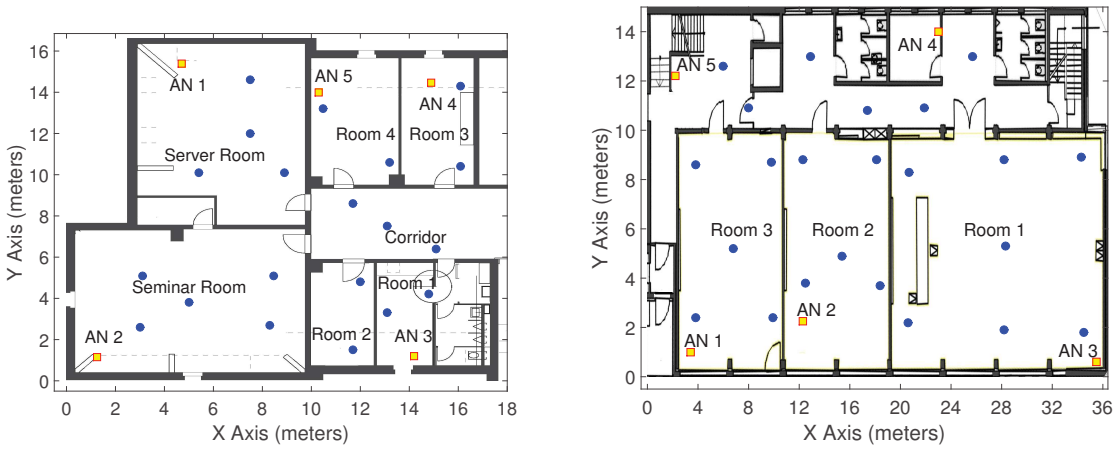
This subsection introduces how to estimate the observation parameter (ranges) and the motion vector (Mv_t) in our proposed particle filter.

3.3.1 Ranging Estimation Process

In order to achieve high ranging accuracy, we adopt the Non-Linear Regression (NLR) model presented in [59] and introduced in Section 2.4.1. The NLR model is defined by Equation 2.3.

Accurate estimation of ranges is a prerequisite to achieve high accuracy on the estimation of the individual likelihood $p(\hat{d}_{j,t} \mid X_t^i)$. Therefore, the raw values of RSSI received from the WiFi sensor are smoothed by approximating the real $RSSI$ value with the mean of the latest four raw RSSI readings.

The first step in ranging estimation is to take the initial measurements, which are aimed to train the environmental parameters α and β required for the NLR model defined in [59]. In our experiments, we defined several stationary points spread over the whole floor plan as shown in Figure 3.3.



(a) Scenario 1

(b) Scenario 2

Figure 3.3: Anchor Nodes distribution and ranging training points (Square Points: Anchor Nodes; Circle Points: Ranging Training Positions)

3.3.2 Enhanced PDR Methods

The mobile target needs to determine the movement of the pedestrian. Therefore, in order to estimate the pedestrian displacement, we use three sensors, the accelerometer, gyroscope, and the geomagnetic field sensor. The displacement of the pedestrian at time t is defined by the motion vector $Mv_t = [Mv_t\theta_t, \ell_t]$. The motion vector Mv_t is passed from the PDR component to the Particle Filter component at instant t when a step of the pedestrian is recognized in the mobile target object. Thereby, step recognition and heading orientation methods are implemented in the PDR component. To develop the step recognition method, we use linear acceleration readings. Figure 3.5 shows linear acceleration behaviour in axes X,Y,Z of the smartphone when a step is executed. Therefore, based on these observations, we develop a step recognition method shown in Algorithm 2.

To estimate the heading orientation, we rely on a digital compass developed from the geomagnetic field and accelerometer sensors embedded in Android smartphones. For further details about heading orientation implementation in Android smartphones, please refer to [5].

Digital compass measures the clockwise angle between the magnetic north and the Y axis of the smartphone at time t. This value is called azimuth (α_t). Therefore, the heading orientation (θ_t) in the local coordinate system can be determined as follows:

$$\theta_t = OffsetX - \alpha_t, \tag{3.10}$$

where *OffsetX* is the clockwise angle between the X axis of our local coordinate system and the magnetic north.

In indoor environments, however, the magnetic fields are usually distorted by electronic devices or ferromagnetic objects. This adversely affects the performance of the digital compass [53]. Therefore, some filtering techniques must be implemented to overcome this problem.

To reduce the noise and cumulative errors from the raw data taken from commodity sensors on smartphones, we apply a low pass filter. Hence, the sequence of values from the accelerometer and magnetic field sensors are smoothed by using the equation 3.11.

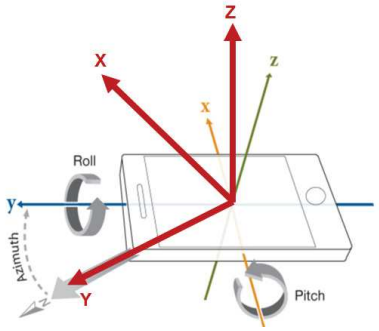


Figure 3.4: Relative Coordinate Systems

$$val_t = val_{t-1} + \kappa(\hat{val}_t - val_{t-1}), \tag{3.11}$$

where κ is a positive smoothing constant ($\kappa < 1$). val_t is the smoothened sensor value at time t. $\hat{val}_t$ is the raw sensor value at time t.

The orientation estimated directly from accelerometer and magnetometer is subject to noise in form of external(i.e. non-gravitational), internal (i.e. acceleration) or magnetic forces that are not caused by the earth magnetic field [33]. Therefore, to overcome this adverse behaviour, we implement a Cosine Direction Matrix (DCM) complementary filter as described in [33]. The key idea of this filter is to combine accelerometer, gyroscope and magnetic field readings to obtain the best guess of the device orientation. The DCM matrix defines the rotation of one coordinate system relative to another. Considering two relative coordinate systems and their respective vectors as shown in Figure 3.4, the relative CDM is defined as follows:

$$DCM = \begin{pmatrix} Y.y & Y.x & Y.z \\ X.y & X.x & X.z \\ Z.y & Z.x & Z.z \end{pmatrix} = \begin{pmatrix} cos(Y,y) & cos(Y,x) & cos(Y,z) \\ cos(X,y) & cos(X,x) & cos(X,z) \\ cos(Z,y) & cos(Z,x) & cos(Z,z) \end{pmatrix}$$

DCM consists of cosines of angles of all possible combination between vectors of the two coordinate systems. Therefore, DCM can be used to represent the relative orientation of the smartphone in the localization system. The key idea of the DCM filter is to determine the angular displacement $d\theta_a$, $d\theta_g$ and $d\theta_m$ from readings of gyroscope, accelerometer and magnetic field sensors respectively. Then, the angular displacement $d\theta$ at time t of the device is obtained by a weighted average as follows:

$$d\theta_t = \frac{(\psi_a d\theta_{at} + \psi_g d\theta_{gt} + \psi_a d\theta_{mt})}{(\psi_a + \psi_g + \psi_m)}, \tag{3.12}$$

where ψ_a, ψ_g, ψ_m are weighting coefficients to be determined experimentally. The update of the DCM filter at t is calculated as follows:

$$I_t \approx I_{t-1} + (d\theta \times K_{t-1}), K_t \approx K_{t-1} + (d\theta \times K_{t-1}), J_t \approx J_{t-1} + (d\theta \times J_{t-1}) \tag{3.13}$$

Although the stride length value can vary along the trajectory, in this work we assume that ℓ is a constant value in order to focus on the tracking algorithm. Nevertheless, the determination of a possible relation between the characteristics of human walking and stride length bias could be the subject of future work.

Algorithm 2 Step Recognition Method

Input : Inertial sensor readings

Output : Step announcement

17 Sense IMUs;

18 **if** *IMUs come from Linear Acceleration sensor* **then**

19 Read $\hat{a}[x,y,z]_t$ vector;

if $\hat{a}[z]_t > threshold$ *and* $\hat{a}[z]_{t-1} < \hat{a}[z]_t$ *and* $\hat{a}[x]_t < \hat{a}[z]_t$ *and* $\hat{a}[y]_t < \hat{a}[z]_t$ **then**

20 Report a Step;

21 **end**

22 **end**

23 Go to step 1;

3.3.3 Graph-based Physical Environment Representation

Physical environmental knowledge does not only provide the movement constraints information (i.e., PDR) but also provides some *likely* areas to move through. Our localization approach defines a discrete structure to replace the conventional floor plan. Thus, all the system states (i.e., indoor positions) can be represented by a discrete set of locations by converting from a continuous state space to a discrete state space. The essential connectivity and accessibility of a complex indoor environment are represented as an undirected graph. Hence, we consider the physical environment

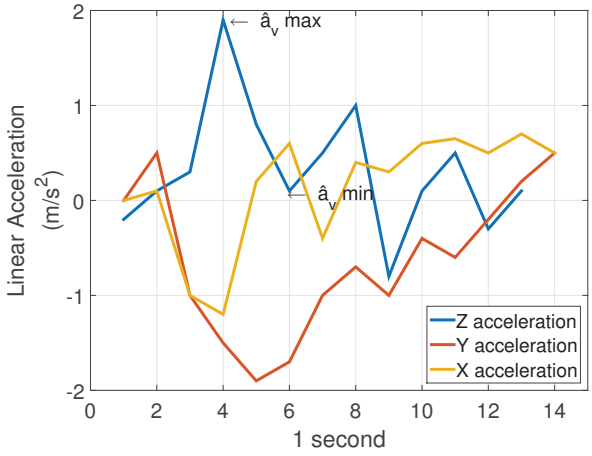


Figure 3.5: Step Recognition, Linear Acceleration Readings

G as a spatial data structure that defines space as an array of nodes arranged in rows and columns. G can be defined as follows:

$$G = (\Upsilon, I, \xi), \tag{3.14}$$

where Υ is a set of nodes, I is a set of features, that defines each node v in Υ. ξ is a set of edges e, that defines connections between nodes v. The grid extraction can be carried out in a few simple steps as follows:

- Specifying the areas where it is permissible to walk in rooms, corridors, etc.
- Building a grid of nodes for each permitted area. Nodes are separated by $0.25m$ from each other in the room and column. Each node corresponds to an element v in Υ.
- Generating the subset of features $i \in I$ for each $v \in \Upsilon$. Each subset i contains three elements related to a v element: coordinates (X, Y) and an identifier of the room to which v belongs.
- Generating set ξ of node connections. Each node is connected with their immediate neighbors in the grid.

Figure 3.6 shows an example of a graphical representation of the discrete state space of our system. Nodes represent positions in the physical environment, whereas edges

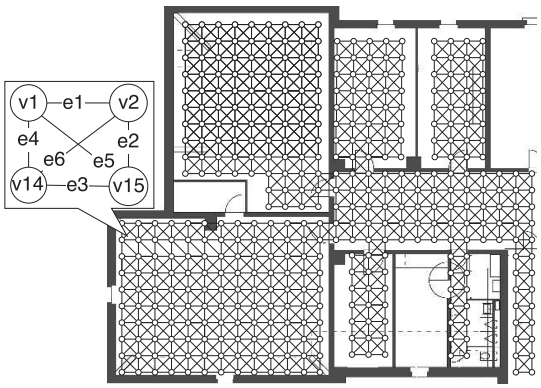


Figure 3.6: Discrete System State Space. Nodes (circles) are interconnected by edges. Edges define the transition model nodes.

define the transition model between nodes. For instance, in Figure 3.6 $e1$ defines a potential transition from node $v1$ to node $v2$ and vise-versa.

3.4 Implementation

We implemented a terminal-based system for accurate indoor tracking. The system comprises two main components: Mobile Target (MT) and Anchor Nodes (ANs). The proposed tracking algorithms are running on the MT. Figure 3.7 presents the overview of the system.

ANs are some commercial Wi-Fi access points deployed at known locations along the area of interest. Positions of ANs are chosen to provide the maximum coverage inside the area of interest. Thus, the location of AN are defined on the boundary corners and the boundary itself.

We have adopted D-Link D-635 and D-Link DAP-2553 as ANs in this work. The beacon period is configured to $100ms$ in ANs.

Mobile targets are some commercial Android smartphones, which support Wi-Fi, inertial, and magnetic field sensors. We have deployed the tracking algorithms in two different models of smartphones, Motorola Nexus 6 and LG Nexus 4. Hereafter, we refer to Motorola Nexus 6 as Mobile Target 1 (MT1) and LG Nexus 4 as Mobile Target 2 (MT2).

In order to save resources in the smartphone, we set the sampling rate of inertial

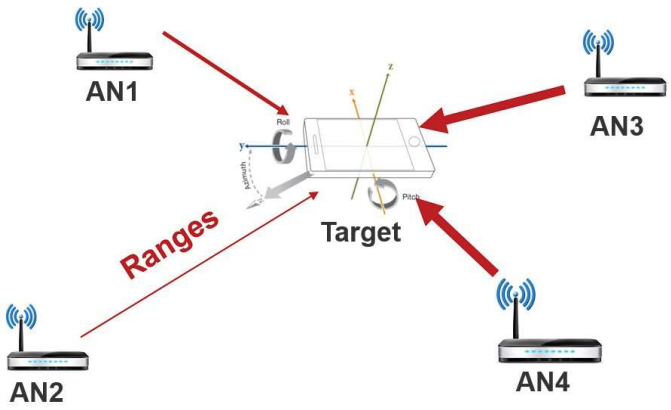


Figure 3.7: Indoor Localization System Overview.

sensors to 14Hz. However, the Wi-Fi sampling frequency is much lower, 3Hz and 4Hz in MT2 and MT1 respectively. It is worth to mention that smartphone models used in this thesis (MT1 and MT2) allow to set up the scanning Wi-Fi radio frequency range. We configured MT1 and MT2 to scan only the 2.4GHz radio frequency range. Thus, by scanning only one radio frequency range, we increase the Wi-Fi sampling frequency on the smartphones. Table 3.1 shows the main characteristics of the mobile targets used in this work.

Additionally, it is necessary to know the floor plan of the area of interest. The system requires information related to restricted areas such as walls. Particles are not allowed to be spread through restricted areas. The system reports the location of the target in real time.

3.5 Performance Evaluation

To evaluate the performance of our proposed system, we conducted a set of experiments in office-like indoor environments. The set of experiments was designed to determine the parameter configuration that leads to the best performance of our indoor tracking system. To do so, we varied the system configuration parameters as follows:

- Target object: we deployed our localization algorithm in two different smartphone models. Therefore, different processing capabilities are taken into account in the experiments. Table 3.1 shows the main characteristics of the smart-

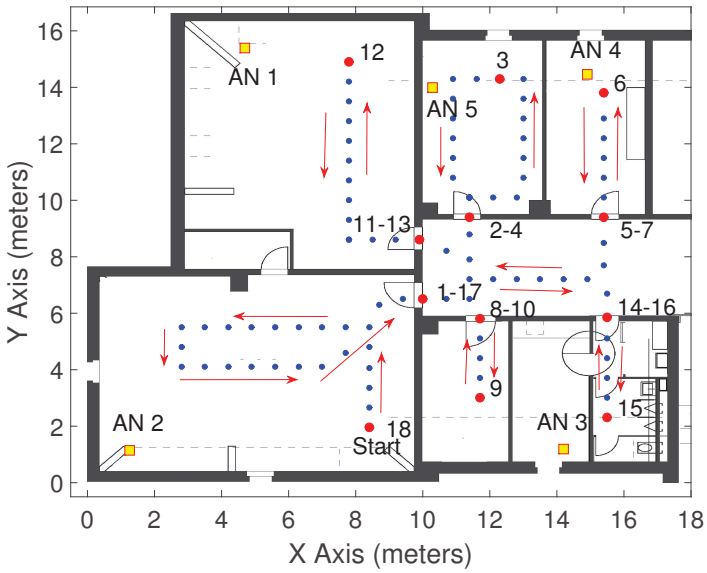


Figure 3.8: Scenario 1, Trajectory 1 and Check point distribution (Square Points: Anchor Nodes; Red Points: Check points; Blue Points: Trajectory path)

phones utilized in our experiments.

- Number of particles: we conducted a set of experiments to achieve the best performance related to the number of particles. The goal of this experiment is to define the number of particles that yields the highest accuracy in the tracking in the process.

- Double resampling method: we validated the tracking accuracy of our system by including the double resampling method proposed in this chapter.

- Number of ANs: we designed a set of experiments by varying the number of ANs from 5 to 4.

- Enhanced PDR methods: we validate the effectiveness of the proposed PDR filters implemented in this chapter.

- Area of interest: experiments were conducted in two different indoor environments along complex and larger trajectories.

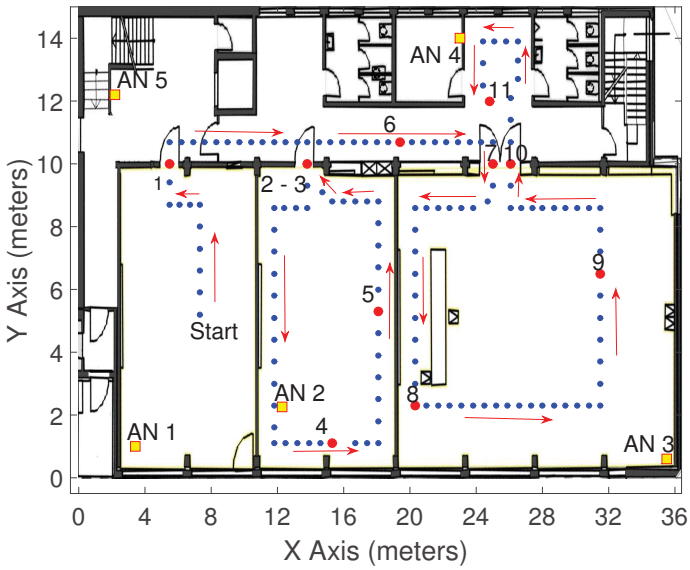


Figure 3.9: Scenario 2, Trajectory 2 and Check point distribution (Square Points: Anchor Nodes; Red Points: Check points; Blue Points: Trajectory path)

3.5.1 Experiment Setup

Experiments were conducted in two buildings of the Institute of Computer Science (INF) at the University of Bern. The first scenario used an area of $288m^2$ and the second scenario $540m^2$. In each scenario, the target mobile device was held in the hand by a person moving along the trajectories as depicted in Figures 3.8 and 3.9. The target mobile device was vertically oriented with the screen facing the roof.

In order to determine the localization errors, some check points were defined along the trajectories. Check points are locations where the ground truth position is known.

Experiments were repeated five times. Therefore, 90 and 55 check points were analyzed in scenario 1 and 2 respectively. The pedestrian was asked to walk through the trajectories and check his current position on the tracking system when he walked through a check point. The difference between positions reported by the tracking system and ground truth positions (check points) are considered as localization error. Figures 3.8 and 3.9 shows the AN distribution, trajectories and check points defined in both scenarios. Blue points define the path of the trajectories, whereas red points define the check points over the moving path. We evaluated also the PDR along these moving paths.

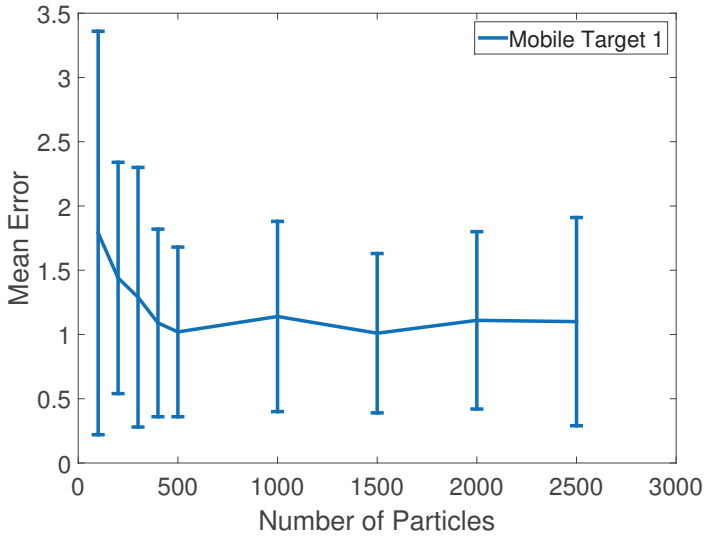


Figure 3.10: Confidence Interval MT1

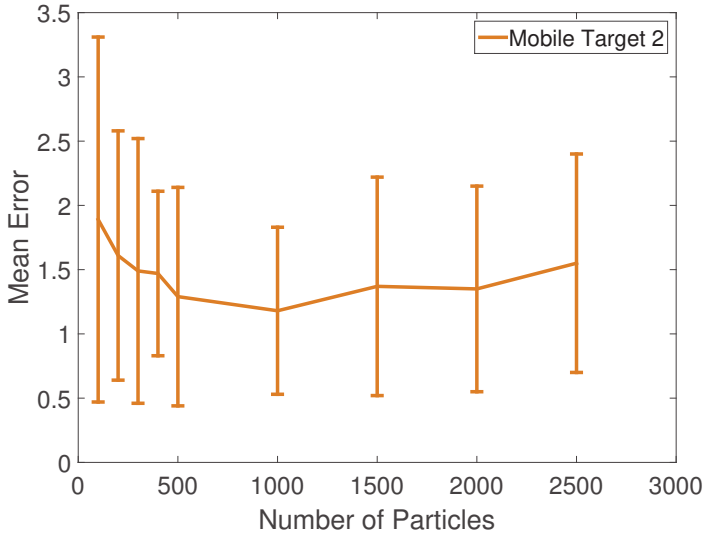


Figure 3.11: Confidence Interval MT2

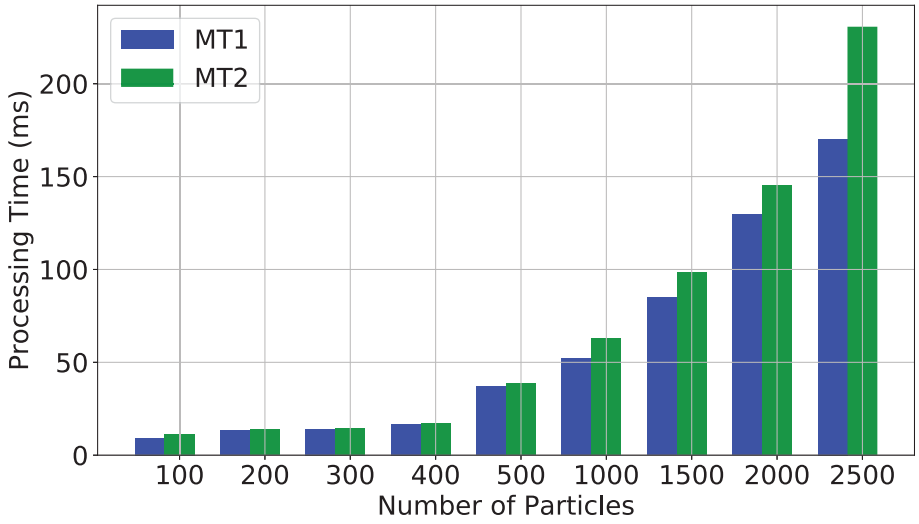


Figure 3.12: Processing Time vs Number of Particles (MT1 and MT2)

Table 3.1: Mobile Targets

Model	Platform
MT 1	**Model:** Motorola Nexus 6; **OS:** Android 5.1.1 **CPU:** Quad-core 2.7 GHz; **RAM:** 3GB **WLAN:**WiFi a/b/g/n **Accelerometer:** Resolution:0.039 Range:19.613 **Magnetometer:** Resolution: 0.150 Range:9830
MT 2	**Model:**LG Nexus 4; **OS:** Android 5.1.1 **CPU:** Quad-core 1.5 GHz; **RAM:** 2GB **WLAN:**WiFi a/b/g/n **Accelerometer:** Resolution:0.001 Range: 39.227 **Magnetometer:** Resolution:0.150 Range:4912

3.5.2 Experiment Results

Experiments were first conducted in scenario 1. Once the configuration that yields to the best performance was defined, we tested our tracking approach in scenario 2. Validation of our proposed asynchronous continuous resampling method was made by using the best configuration parameters defined in the experiments conducted in scenario 1.

Performance vs Number of Particles

This set of experiments was conducted in scenario 1 with 5 ANs. Details about the environment configuration can be seen in Figures 3.8 and 3.9. The different choices regarding the number of particles are 100, 200, 300, 400, 500, 1000, 1500, 2000, 2500. Figures 3.10 and 3.11 show the confidence intervals resulting from the experiments conducted in both MTs. Figure 3.12 shows the processing time regarding the number of particles. Location accuracy can be theoretically boosted by using more particles, that is, the more particles, the better is the accuracy [50]. However, we can see in our experiments that after a certain number of particles the tracking errors remain almost constant. Thus, the accuracy in both MTs is slightly improved after a certain number of particles. In the case of MT1, we can see that the minimum mean error $1.01m$ is achieved with 1500 particles which spend approximately $51.9\mu s$ of processing time. In MT2, the minimum mean error $1.18m$ is accomplished with 1000 particles, which take $49.81\mu s$ of processing time. To explain this behaviour, we look at the negative influence produced by increasing processing time in real-time systems. The efficiency of real-time systems depends not only on the precise results but also on the time to get these results. In real-time localization, high processing time could lead the system to stay processing a position while the ground truth position is changing. Therefore, clearly in real-time localization applications, processing time influences the accuracy performance of the system. We noticed that the average processing time seems to grow exponentially with the number of particles (see Figure 3.12). Due to the negative influence of the exponential growth of the processing time, there is no more performance improvement when a certain particle number is used.

Table 3.2: Performance vs Number of ANs

Configuration	Mean error	S.D	90% Acc.
MT1 1500 Ptc. (5ANs)	1.01m	0.62m	1.7m
MT2 1000 Ptc. (5ANs)	1.18m	0.65m	2.1m
MT1 1500 Ptc. (4ANs)	1.16m	0.7m	2.3m
MT2 1000 Ptc. (4ANs)	1.43m	0.86m	2.7m
PDR	8.6m	5.49m	17.2m

Performance vs Number of ANs

In this experiment, we selected the number of particles that yields the best performance of our localization approach. Hence, the chosen number of particles in this experiment was 1500 for MT1 and 1000 for MT2. Figure 3.13 shows CDF (Cumulative Distribution Function) of positioning errors of our proposed indoor tracking approach in scenario 1. Figure 3.14 shows CDF of tracking errors of our approach and the PDR system. Since PDR performance is independent of the number of ANs and number of particles, the CDF of PDR summarizes all the experiments in both scenarios that were utilized in this work. Table 3.2 summarizes the mean tracking error, standard deviation and 90% accuracy.

It is well known that PDR is prone to accumulated errors because it estimates the current location of the target by integrating the relative movement from the previous locations [50]. Therefore, the larger the trajectory is, the bigger are the average of localization errors in a PDR system.

Experiment results show that our proposed indoor tracking approach achieves more stable and higher accuracy than a PDR system. In order to further evaluate our tracking approach, we evaluate the system along large and complex trajectories. The experiment results show 17.2m for 90% accuracy using PDR because of accumulative errors. However, our tracking approach efficiently deals with the accumulative errors even when the number of ANs is decreased to 4. Our tracking approach achieves around 1.7m for 90% accuracy, which outperforms PDR-based localization system by around 90.1% along trajectory 1. The mean error of our tracking approach is 1.01m, which is 88.2% smaller than PDR. Standard deviation is 0.62 which is 88.7% smaller than PDR. Thus, our proposed approach outperforms PDR considering accuracy and stability.

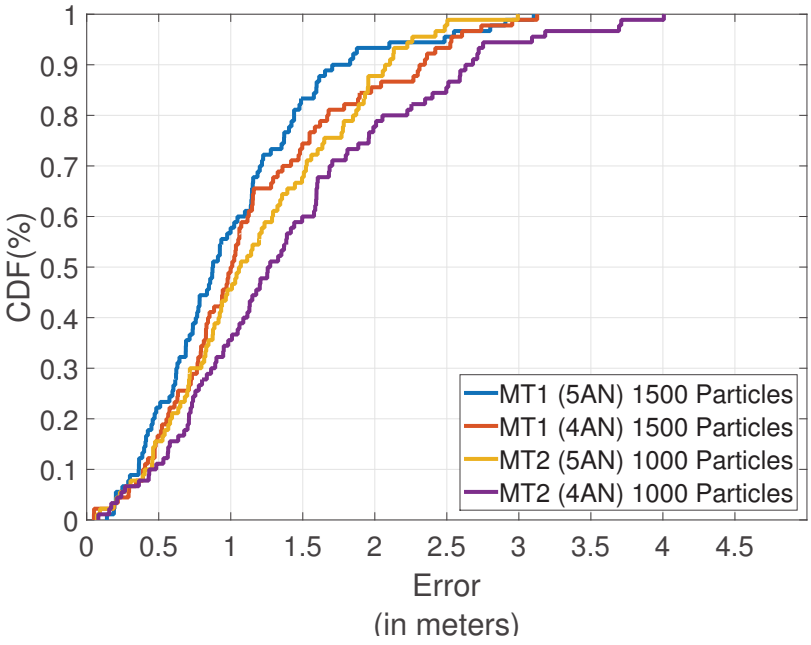


Figure 3.13: Configuration Parameters

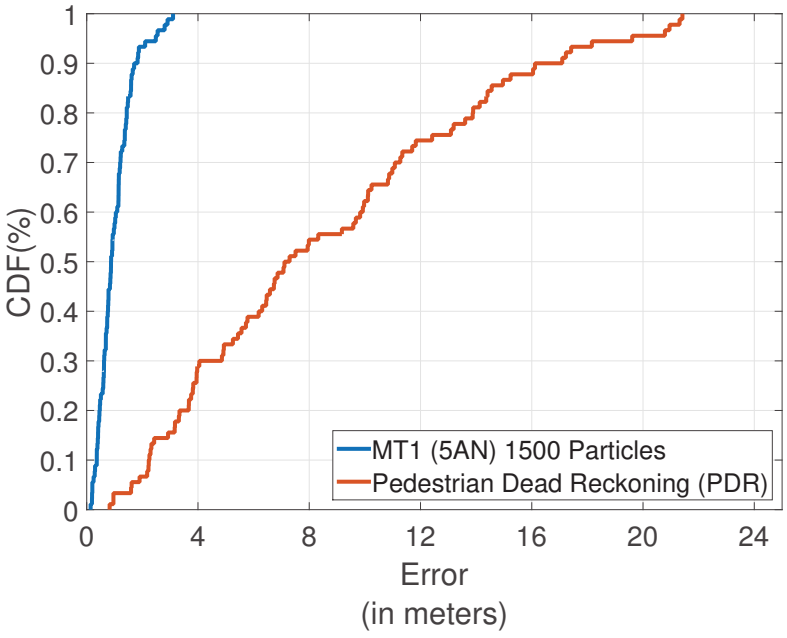


Figure 3.14: Particle Filter vs PDR, scenario 1

Table 3.3: Double resampling and single resampling methods, scenario 1

Tracking Approach	Mean error	S.D	90% Acc.
Double Resampling	1.01m	0.62m	1.7m
Single Resampling	1.61m	1.02m	3m

Performance vs Resampling Methods

Figure 3.15 shows the CDF of positioning errors for the tracking algorithms. As mentioned before, we refer double resampling to the asynchronous continuous correction phase. The single resampling approach does not include the asynchronous continuous correction phase in the localization approach.

Our tracking approach along double resampling method achieves higher accuracy and more stable performance compared to the single resampling approach. Table 3.3 summarizes the average of tracking errors, standard deviation and 90% accuracy. Our tracking approach along double resampling method achieves around $1.7m$ for 90% accuracy, whereas the single resampling method achieves around $3m$. The mean error of the double resampling method is $1.01m$, whereas the mean error of single resampling is $1.61m$. Standard deviations are $0.62m$ and $1.02m$ in double and single resampling method respectively.

Based on these results, we can find the following observations. First, our tracking approach outperforms the single resampling approach by around 43.3% considering 90% accuracy. Second, the mean error of our tracking approach is 37.3% smaller than the single resampling method. Third, standard deviation is 39.2% smaller than the single resampling method. Therefore, experiment results show that because of the continuous resampling method, our tracking approach is able to faster correct the mobile target position along the moving paths compared to single resampling. The double resampling approach outperforms the single resampling approach considering accuracy and stability.

Performance vs Area of Interest

This experiment validates the environment independence of our approach. Thus, we chose a second scenario to deploy the tracking system. As mentioned in previous sections, scenario 2 is placed in different buildings at the University of Bern. The area of interest is 46.7% bigger than the area in scenario 1. However, we set up our approach with the configuration that achieved the best performance on experiments

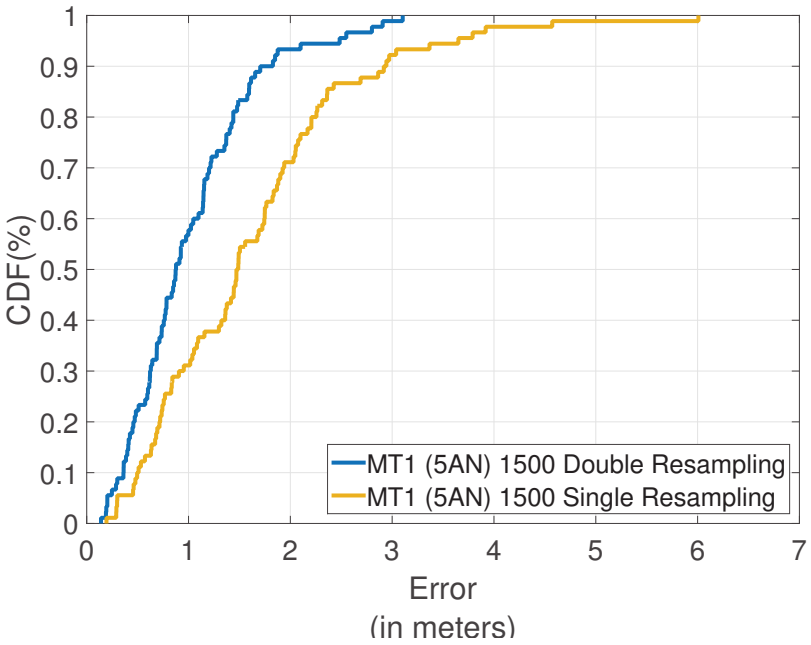


Figure 3.15: Tracking error CDF on asynchronous continuous resampling (double resampling) vs single resampling method

executed in scenario 1. That is, the tracking algorithms are deployed on MT1 with 1500 particles and 5 ANs distributed along the area of interest.

Figure 3.9 depicts scenario 2 and trajectory 2. Table 3.4 summarizes the mean tracking error, standard deviation and 90% accuracy. Figure 3.16 depicts the CDF of positioning errors for our tracking approach tested in scenario 2. Our tracking approach using the double resampling method achieves around $3m$ for 90% accuracy, which outperforms PDR by around 86.36% along trajectory 2. The mean error is $1.6m$, which is 88.28% smaller than PDR. Standard deviation is $0.9m$, which is 84.62% smaller than the PDR system.

Based on these observations, we can highlight that despite the bigger area of interest, our proposed tracking approach achieves high accuracy and stable performance. However, the mean error and standard deviation were slightly increased compared to experiments in scenario 1. This reflects that the density, positions, and coverage of ANs along the area of interest influence the tracking precision.

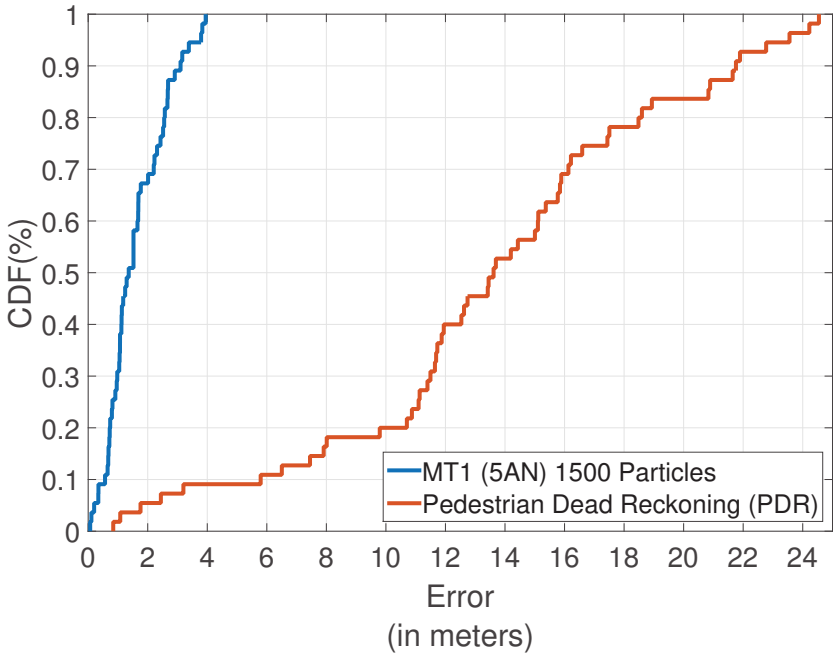


Figure 3.16: Tracking error CDF on scenario 2.

Table 3.4: Double resampling and PDR methods, scenario 2

Tracking Approach	**Mean error**	**S.D**	**90% Acc.**
Double Resampling	1.6m	0.9m	3m
PDR	13.66m	5.85m	22.3m

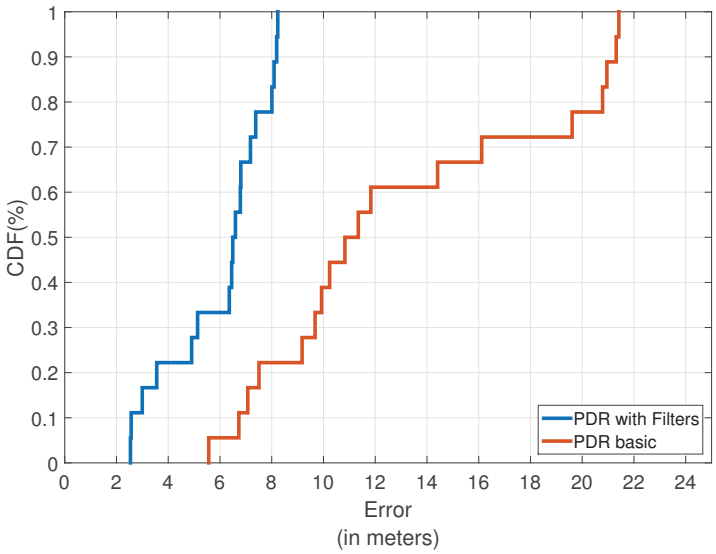


Figure 3.17: PDR algorithms comparison.

3.5.3 Pedestrian Dead Reckoning Filters

In order to validate the performance of the proposed PDR algorithm along with filters, we compare the performance with the basic PDR system used in [20]. The testing trajectory starts at the point marked with the label *start*. This point is also the last check point at the end of the trajectory. Figure 3.17 shows the CDF of positioning errors for the PDR algorithms. Hereafter, we will refer to the PDR algorithm with filters as the PDR filter.

The PDR filter algorithm achieves higher accuracy and more stable performance compared to the basic PDR algorithm. Table 3.5 summarizes the average of tracking errors, standard deviation and 90% accuracy. The PDR filter algorithm achieves around $8.15m$ for 90% accuracy, whereas the basic PDR algorithm achieves around 21.6%. The mean error of the PDR filter algorithm is $6.0152m$, whereas the mean error of the basic PDR is $13.027m$. Standard deviation is $1.9461m$ and $5.5872m$ in PDR filter and PDR basic respectively.

Based on these results, we can find the following observations. First, the PDR filter algorithm outperforms the basic PDR algorithm by around 62.26% considering 90% accuracy. Second, the mean error of the PDR filter algorithm is 53.82% smaller than the basic PDR algorithm. Third, the standard deviation of the PDR filter algorithm

Table 3.5: PDR algorithms performance

Algorithm	**Mean Error**	**S.D.**	90% **Acc.**
PDR with filters	6.0152m	1.9461m	8.15m
Basic PDR	13.0279m	5.5872m	21.6m

is 65.17% smaller than the PDR basic algorithm. Therefore, experiment results show that the PDR algorithm along the low pass and CDM filter is able to improve the localization accuracy. The PDR filter algorithm outperforms the basic PDR algorithm for accuracy and stability.

3.6 Conclusions

In this chapter, we exploited an enhanced particle filter approach to fuse radio signals, inertial sensors and physical information of the environment, to achieve high localization and tracking accuracy in complex indoor scenarios using commodity smartphones. Additionally, this chapter presented an asynchronous continuous correction phase that is able to tackle the low sampling rate problem of Wi-Fi sensors on the smartphone side. This phase is asynchronously executed whenever new Wi-Fi RSSI information is available. The localization approach is formulated in a discretized graph-based representation of the indoor environment. Moreover, we integrated an enhanced PDR method by considering magnetic field and angular velocity measurements to further improve the heading orientation estimation.

We evaluated our localization system in two complex large trajectories in different indoor scenarios. Experiments show that our approach can achieve an average tracking error of $1.01m$ and 90% accuracy is $1.7m$. Thus, our tracking approach is more accurate and stable than pedestrian dead reckoning and the commonly used particle filter with single resampling. Furthermore, our proposed approach enables tracking in the smartphone side without the assistance of any additional external server.

Despite particle filter-based localization methods can achieve high accuracy, there are some common issues to overcome. These issues can lead the system non-functional or non-efficient. Two of the most common issues are related to localization failure problems such as the global localization problem and the kidnapped-robot problem. Thus, in the next chapter, we focus on solving localization failures problems in particle filter-based localization systems.

4

Failure Recovery Mechanism for Monte Carlo Localization based Systems

4.1 Introduction

The idea of robust indoor localization entails providing methods to prevent and recover Monte Carlo Localization Systems (MCLS) from localization failure problems. In MCLS, two of the most common localization failure problems are the global localization problem and the KRP. The former happens when the localization system starts. Here, the initial position of the target is unknown. The target is located somewhere in the environment without any knowledge of its position.

Unlike the global localization problem, the KRP occurs during system operation. Thus, the KRP appears when a well-localized target in operation moves to some arbitrary locations, while the target is not aware of this. Therefore, the target device might firmly believe to be somewhere else at the time of the kidnapping.

Most of the state-of-the-art localization approaches cannot be guaranteed never to fail

[102]. Therefore, the ability to recover from failures is essential for truly autonomous MCL systems. In this chapter we focus on addressing localization failure problems in MCL based systems. We propose an efficient method to recover the localization system from localization failures. Our recovery method uses machine learning techniques, which provide zone level localization. We consider a zone as a subarea inside the target area (e.g., rooms, corridors). Thus, we incorporate zone level localization results in our enhanced particle filter to faster recover the system from localization failures.

Since our proposed recovery failure method relies on zone level localization results, we also focus on providing enhanced machine learning methods to achieve high zone level localization performance. Typically, ensemble machine learning approaches achieve better predictive performance compared to individual models. Ensemble learning methods aim to improve prediction performance by combining several individual machine learning techniques into one predictive model. Thus, in this chapter we also propose two novel ensemble machine learning models to provide zone level localization. The proposed ensemble learning models combine conceptually different individual machine learning algorithms to predict class labels, such as rooms, zones. The main contributions included in this chapter are summarized as follows.

- We propose an ensemble learning method by combining conceptually different individual machine learning algorithms to predict class labels (i.e., rooms, zones, landmarks). This combination is made by applying concepts of conditional probability and evidences about the prediction performance of individual predictors. We call this approach Conditional Probability Ensemble Learning Method (COND).
- We propose an ensemble learning method based on a HMM. We call this method HMM-based Ensemble Learning Method (HMM-d). HMM-d integrates ambient signal information and coarse-grained floor plan information in a HMM.
- We propose an efficient method to recover the system from localization failures. Our method uses a machine learning approach, which provides zone level localization. Thus, we integrate zone level localization in the enhanced particle filter to recover the system from localization failures, such as the global localization problem and KRP.

The rest of the chapter is organized as follows. Section 4.2 presents COND method. Section 4.3 introduces HMM-d method. In Section 4.4 we present the proposed

localization failure recovery mechanism. Implementation details about COND, HMM-d, and the failure recovery methods are presented in Section 4.5. Section 4.6 presents the evaluation results of our proposed approaches. Section 4.7 concludes the Chapter.

4.2 Ensemble Conditional Probability Method (COND)

COND is based on the concept of conditional probability. In probability theory, conditional probability is a measure of the probability of an event considering some evidences. Thus, the probability of being located at a zone considering some evidences can be written as follows:

$$P(c_i \mid l_1, l_2, ..., l_n) = \frac{P(l_1, l_2, ..., l_n \mid c_i) \cdot P(c)}{P(l_1, l_2, ..., l_n)}, \tag{4.1}$$

where c_i is the zone identifier (i.e., the class) and l_i is the *i-th* evidence provided by the *i-th* machine learning prediction model.

Equation 4.1 can be solved assuming conditional independence of events l_i given the event c_i. Conditional independence means that if some piece of information is known, the probability of other events become independent. For the zone level localization problem, our assumption is that the probability of obtaining the outcome l_i becomes independent if the value of c_i is known. Thus, Equation 4.1 can be written as follows:

$$P(c_i \mid l_1, l_2, ..., l_n) = \frac{P(l_1 \mid c_i) \cdot P(l_2 \mid c_i) \cdot ... \cdot P(l_n \mid c_i) \cdot P(c_i)}{P(l_1, l_2, ..., l_n)} \tag{4.2}$$

Considering that individual predictors are independent from each other,

$$P(c_i \mid l_1, l_2, ..., l_n) = \frac{P(l_1 \mid c_i) \cdot P(l_2 \mid c_i) \cdot ... \cdot P(l_n \mid c_i) \cdot P(c_i)}{P(l_1) \cdot P(l_2) \cdot ... \cdot P(l_n)} \tag{4.3}$$

Therefore, the prediction of the zone z can be calculated as follows:

$$z = \arg\max_{c} \frac{P(c) \cdot \prod_{i=1}^{n} P(l_i \mid c)}{\prod_{i=1}^{n} \sum_{j=1}^{m} P(l_i \mid c_j) \cdot P(c_j)} \tag{4.4}$$

where z represents the predicted class, n is the number of evidences given by n machine learning individual predictors, and m is the number of landmarks (i.e, classes).

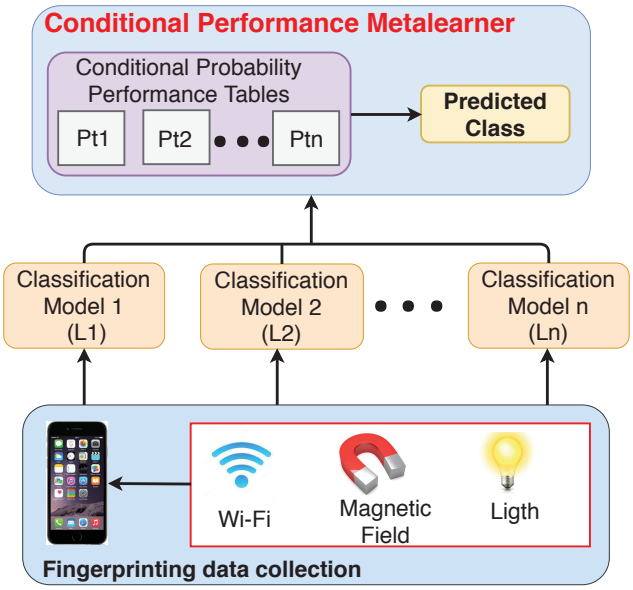


Figure 4.1: COND Learning Method Architecture

Figure 4.1 depicts an overview of the COND learning method architecture. The key idea of COND is to combine conceptually different individual machine learning models to predict class labels (i.e., zones, rooms, landmarks). This combination is made by applying conditional probability concepts and information about the individual prediction performance of each machine learning model. Thus, the outcome of each individual machine learning predictor can be regarded as an evidence l_i in COND. The prediction performance given the knowledge of the ground-truth class label defines the probability of the occurrence of an evidence given the ground-truth label class $P(l_i \mid c)$. From the implementation view, COND uses Conditional Performance Tables (CPT) to store $P(l_i \mid c)$ of each individual machine learning model. Thus, $P(l_i \mid c)$ and l_i are used in the prediction process. Considering the performance of each individual learning model, COND balance out their strengths and weaknesses to improve the prediction performance.

4.3 Ensemble HMM-based Method (HMM-d)

In this section, we introduce an ensemble learning model by combining HMM with discriminative machine learning methods. Figure 4.2 shows the probabilistic parameters of the proposed HMM-d method. It is worth to notice that this model is suitable for any zone detection (i.e., any subarea in the area of interest). Hereafter, we will

refer to room as zone. The proposed HMM-d method is based on the concept of Markov localization [31], which can be described as estimating the state of HMM with controllable state transitions. For the localization problem, we refer to zones as states of the HMM. Thus, our HMM is specified by the following components:

- A set of states $Z = \{z_1, ..., z_n\}$, where z_l is the identifier value of the zone l. Therefore, the hidden state z_l at time t can be represented by the discrete random variable $z_{l_t} \in Z$.

- A transition probability matrix A,

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n} \\ a_{2,1} & a_{2,2} & ... & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & ... & a_{n,n} \end{pmatrix},$$

  where $a_{k,l} \in A$ is the likelihood of moving from zone z_k to zone z_l. Thus, A is a square matrix of order n.

- A set of observations C,

$$C = \{(c_1, ..., c_m)_1, ...(c_1, ..., c_m)_{n^m}\}, \tag{4.5}$$

  where $c_k \in Z$ is the prediction outcome of the k-th constituent individual machine learning algorithm. Thus, C is a set of ${}^nP_m$ permutations with repetition allowed, where n is the number of zones and m is the number of individual machine learning algorithms that constitute the HMM-d method. We define $(c_1, ..., c_m)_l \in C$ as q_l. The observation q_l at time t can be represented by the random variable $q_{l_t} \in C$.

- A matrix B of observation probabilities. B is named the emission probability matrix.

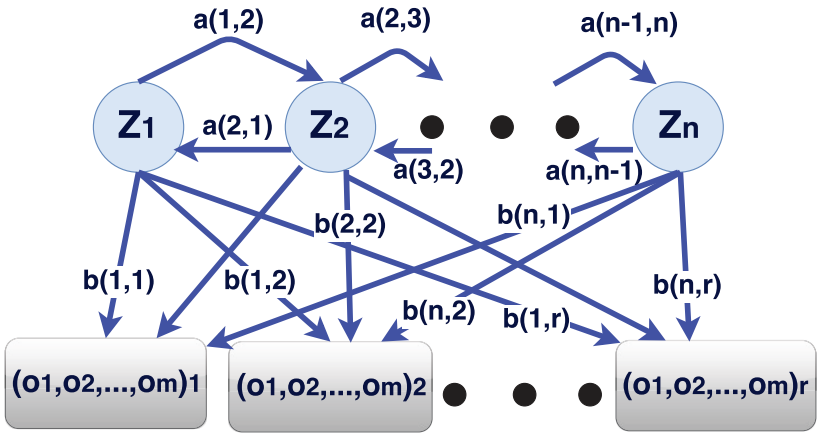


Figure 4.2: Probabilistic parameters of the proposed HMM-d Method.

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,n^m} \\ b_{2,1} & b_{2,2} & \dots & b_{2,n^m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \dots & b_{n,n^m} \end{pmatrix},$$

where $b_{k,l}$ is the likelihood of observing $q_l \in C$ at zone z_k.

- An initial probability distribution over zones $\pi = \pi_1, ..., \pi_n$, where π_i is the probability of being located in zone i.

The individual learning methods only rely on the latest observed fingerprint (e.g., Wi-Fi RSS and MF readings) for localization, which may produce incorrect prediction results. However, HMM-d can be used to integrate zone transition information and the current observed information (e.g., current observed fingerprint) to improve prediction results.

In any model with hidden variables (e.g., HMM), the task of determining the sequence of variables (e.g., zones) that is the underlying source of some sequence of observations is named the decoding task. Thus, given a sequence of observations $y_{t-i}, ..., y_{t-1}, y_t$, and a settle model HMM $\lambda = \{\pi, A, B\}$, the sequence of hidden states $x_{t-i}, ..., x_{t-1}, x_t$ can be estimated by employing the Viterbi algorithm [39].

4.3.1 Transition Probabilities

The transition probabilities express the likelihood of moving from one state (i.e., zone) to another. Zones must be defined beforehand. Connections among zones in the coarse-grained floor plan determine the transition probabilities. Therefore, the transition probability matrix can be written as follows:

$$T = \{a_{ij} = P(x_{t+1} = z_j \mid x_t = z_i)\}, \tag{4.6}$$

where A is a $n \times n$ matrix, a_{ij} represents the transition likelihood between zone z_i to zone z_j. Therefore, $\sum_{j=1}^{n} a_{ij} = 1$.

4.3.2 Emission Probabilities

The emission probability is the likelihood of producing a particular set of observations y_j at zone z_i. Therefore, the emission probability matrix can be written as follows:

$$B = \{b_{ij} = P(y_j \mid z_i)\}, \forall\, y_j \in C \wedge z_i \in Z, \tag{4.7}$$

where $y_j = (c_1, c_2 ... c_m)_j$ and c_i is the zone prediction result from the i-th individual learning method. Since individual machine learning methods are different and independent of each other, it is reasonable to assume that their outcomes are conditionally independent. Therefore, b_{ij} can be written as follows:

$$b_{ij} = \prod_{n=1}^{n} P(c_j \mid z_i)_n, \tag{4.8}$$

where $P(c_j \mid z_i)_n$ is the probability of predicting c_j at zone z_i by the n-th individual discriminative learning method. Therefore, $P(c_j \mid z_i)_n$ represents the sensitivity of the individual learning method n at zone i. Thus, $P(c_j \mid z_i)_n$ can be written as follows:

$$P(c_j \mid z_i)_n = \frac{TP_n}{TP_n + FN_n}, \tag{4.9}$$

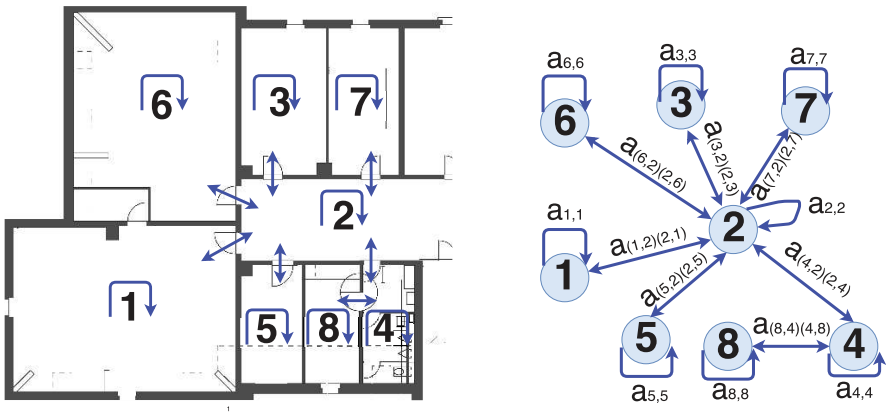
where TP_n and FN_n are the true positive and false negative rate of the n-th individual discriminative learning method.

Algorithm 3 Asynchronous Continuous Resampling

Input : Floor Plan, $R\hat{S}SI$, current zone, failure alert

Output : Particle's corrected weights, failure recovery

if *new* $R\hat{S}SI$ *reading is available* **then**
    current zone=machine learning($R\hat{S}SI$)
    foreach *Particle* P_t **do**
        Check:P_t.zone ==current zone
    end
    if *There is not particle* P_t*.zone==current zone* **then**
        failure alert counter++
    end
    if *There is a particle* P_t*.zone== current zone* **then**
        Initialize failure alert counter
    end
    if *failure alert counter== failure threshold* **then**
        Resample (zone recognition accuracy)% particles in current zone.
        Resample (1-zone recognition accuracy)% particles entire environment.
    end
    Determine $R\hat{S}SI$ mean ($mRSSI$) of the latest 4 $R\hat{S}SI$ readings:
    $mRSSI = \frac{\sum_{k=0}^{3} RSS_{t-k}}{4}$
    foreach AN_j **do**
        calculate $d_j = \alpha_j \cdot e^{\beta_j \cdot mRSSI_j}$
    end
    Calculate individual likelihood: $p(d_j \mid X_t^i)$; *Normalize weights*
    foreach *Particle* P_t **do**
        $W_t = \hat{W}_t / \sum_{n=1}^{N} \hat{W}_n^i$
    end
    Resample particles P_t based on systematic resampling method.
end
Go to 1



(a) Zone definition and graphical transition model

(b) Transition model for HMM-d

Figure 4.3: Zone definition and transition model for HMM-d.

4.4 Localization Failure Recovery Method

As mentioned before, the proposed localization failure recovery method relies on zone localization results. Algorithm 3 describes the proposed failure recovery method. After obtaining zone localization results, localization failures problems are covered by two stages: first, failures recognition; second recovering from failures. We incorporate machine learning approaches in the asynchronous continuous correction phase of the enhanced particle filter introduced in Chapter 3. For failure recognition, the machine learning algorithm first evaluates the current zone where the mobile target device is placed, then the zone reported by each particle is evaluated. If there is not at least one particle in the same zone reported by the machine learning algorithm, a failure alert is launched. To recover the system from localization failures, our proposed method follows three basic steps:

- Current zone equally distributed particles resampling. A percentage of particles is spread over the zone reported by the machine learning algorithm. This percentage of particles is the same as the zone recognition accuracy.

- The particles that perform the resampling operations are equally distributed over the entire target area.

- The weight of each particle is evaluated based on range likelihood information.

To summarize, the proposed Correction phase is aimed to accomplish three objectives. The first goal, as introduced in Chapter 3, is to make a continuous correction of the level of influence that each particle contributes to the determination of the state of the system. The second objective is to recognize localization failures. The third goal is to recover the system from localization failures.

4.5 Implementation

This section presents implementation details of COND, HMM-d, and the proposed localization failure recovery methods.

4.5.1 Zone Level Localization Methods Implementation

For COND and HMM-d methods, fingerprint readings are measured by the target mobile device and received by an app. We then perform the offline training process to build the machine learning models in a PC. The built machine learning models are then optimized and transferred to the app on the mobile target device for online experiments. To reduce the negative impact of environmental changes and different hardware, we use differential Wi-Fi RSS instead of absolute raw values [108].

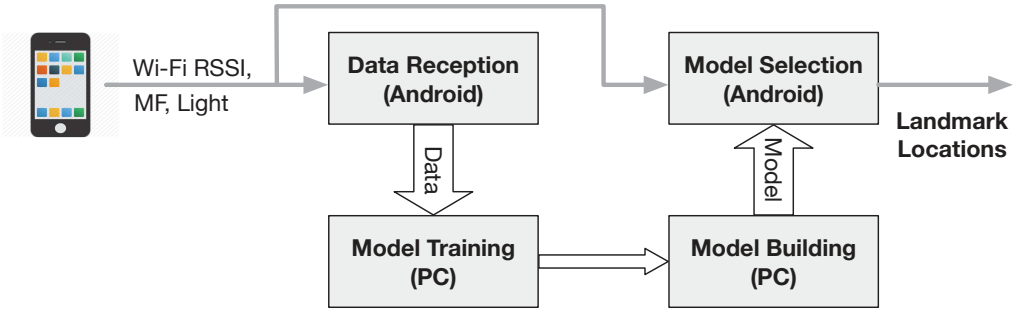


Figure 4.4: COND and HMM-d Components and Data Flow.

Figure 4.1 shows the data flow and the different components of COND and HMM-d methods. The device's sensors measure the magnetic field in the device's coordinate system. As the user walks around, the orientation of the device may change all the time. To avoid this problem, magnetic field values must be translated from the device's coordinate system to a global coordinate system, such as the earth' coordinate system. We use rotation matrix to translated magnetic field values to the earth' coordinate system. Android includes a library to perform this translation. Details can be found

in [5].

Light sensors might also be helpful to improve zone localization performance. For instance, a room facing a window will clearly be brighter than one surrounded by walls only. However, these assumptions are not stable, as the illuminance level might change over time. To deal with this instability, we consider the light illuminance differences instead of absolute values. Thus, we measure the illuminance level of zone 1, then difference values are computed for the remaining zones with respect to zone 1.

Parameters of learning algorithms are optimized from training data. Additionally, certain algorithms also have parameters that are not optimized during the training process. These parameters are called hyperparameters, which have significant impact on the performance of the learning-based algorithm. Therefore, we use a nested cross validation technique [76] to adjust them. The nested cross validation technique defines inner and outer cross validation. Inner cross validation is intended to select the model with optimized hyperparameters, whereas outer cross validation is used to obtain an estimation of the generalization error. Ten-fold cross validation was applied on both inner and outer cross validation. The classifiers were optimized over a set of hyperparameters.

We performed experiments on the third floor of the Computer Science building at the University of Bern, as shown in Fig. 4.5. To build the COND and HMM-d fingerprint databases, we ask a person to walk randomly around each room holding the phone in his/her hand. The data collection rate is only constrained by computational capabilities of the smartphones' Wi-Fi interface. Thus, in our experiments every data measurement was collected at a rate of 3 entries/second. Because our approach does not need any survey point, the time needed to build the landmark fingerprint database is proportional to the number of collected instances multiplied by the instance collection rate.

Implementation and data collection processes to build the fingerprint databases of COND and HMM-d were performed as part of bachelor thesis projects [78], and [111] respectively. COND and HMM-d implementation and offline training processes are different and independent from each other. Thus, in the following paragraphs, we present implementation and evaluation details of COND and HMM-d separately.

COND Method Implementation

As shown in Figure 4.5, we define 14 wall separated areas as landmarks in our experiments (i.e., 14 class labels). Hereafter, we refer to these areas as rooms. In our experiments, we do not need to know the exact locations of the Wi-Fi APs, while only the fingerprints of Wi-Fi RSSI, MF readings, and illuminance level readings are recorded during the data collection procedure.

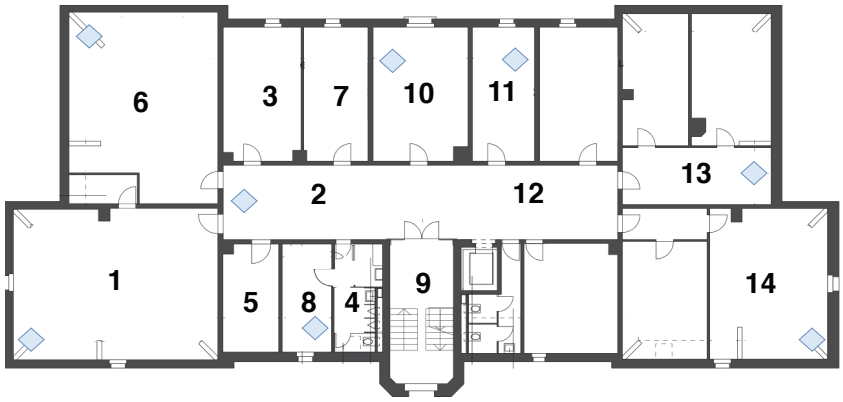


Figure 4.5: COND Experiment Scenario, zone definition and ANs distribution (Diamond blue points: Anchor Nodes).

To ensure conditional independence between the individual learning methods in COND, we setup three completely different discriminative machine learning techniques for the zone prediction method. KNN [79], Multilayer Perceptron (MLP) [110] and the SVM [72] machine learning algorithms were selected. We optimized the global blend percentage ratio hyperparameter for KNN, the kernel type function for SVM, and the number of hidden layers and neurons per layer for MLP. Based on the parameter optimization process, we established the optimal hyperparameter values for the classifiers as follows: global blend percent ratio of 30% for KNN, single order polynomial kernel, 646 support vectors, $c = 1$, and $\gamma = 0.0$ for SVM, and single hidden layer with 10 neurons for MLP.

To build the fingerprint database, we collected 17569 data points in total, which were equally distributed along the whole area in each room. Collecting the training dataset takes around 75 minutes. With the collected data, we build models with different data: the first one builds the fingerprint using only collected Wi-Fi RSS data, the second one using Wi-Fi RSS together with MF readings, and the third one using Wi-Fi RSS, MF readings, and illuminance level readings.

As mentioned, the data collection process to build the fingerprint database was performed as part of a bachelor thesis project [78].

HMM-d Method Implementation

We performed experiments on the left part of the third floor of the Computer Science building at the University of Bern. Figure 4.3a presents an overview of the target area. It is necessary to have coarse-grained information about the area of interest. The HMM-d method requires information related to zone distribution and physical connections among zones (i.e., zone transition information). Zone information is also included in the coarse-grained floor plan (i.e., how the area of interest is split in zones).

Figure 4.3 shows the zone definition and transition model in our HMM-d method. The basic assumption to compute matrix A is that the likelihood of staying at the same zone is higher than the likelihood of going to another one. Thus, the transition probability matrix A was empirically defined as follows:

$$A = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.4 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0 \\ 0 & 0.4 & 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0.6 & 0 & 0 & 0 & 0.2 \\ 0 & 0.4 & 0 & 0 & 0.6 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0.4 & 0 & 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0.6 \end{pmatrix},$$

To ensure conditional independence between the individual learning methods in HMM-d, we setup three completely different discriminative machine learning techniques for the zone prediction method. KStar [26], Multilayer Perceptron (MLP) [110] and the J48 decision tree [83] machine learning algorithms were selected.

The classifiers were optimized over a key of hyperparameters. We optimized the global blend percentage ratio hyperparameter for KStar [26], the confidence factor for J48 [83], as well as number of hidden layers, number of neurons per layer, and activation function for MLP [110]. Based on the parameter optimization process, we established the optimal hyperparameter values for the classifiers as follows: global

percent ratio of 30% for KStar, single hidden layer with 10 neurons and sigmoid activation function for MLP. For J48, the confidence factor was configured to 0.25.

To build the fingerprint database, we collected 10000 fingerprints in total, which were equally distributed along the whole area in each room. As mentioned, the data collection process to build the fingerprint database was performed as part of a bachelor thesis project [111].

4.5.2 Localization Failure Recovery Method Implementation

The proposed failure recovery method is implemented in terminal-based system architecture. Algorithms are implemented in java language for Android smartphones [5]. The system comprises two main components: mobile target (MT) and Anchor Nodes (ANs). MT are some commercial Android smartphones, which support Wi-Fi, inertial, and magnetic field sensors. The localization failure recovery algorithms are running on the MT. Localization failure experiments were performed on the left part of the third floor of the Computer Science building at the University of Bern. Figure 4.3a presents an overview of the target area.

4.6 Performance Evaluation

4.6.1 Performance Evaluation COND Method

This section presents and discusses zone level localization results of the COND method. We divided the area of interest into two scenarios. We defined scenario 1 as the left part of our area of interest, which covers zones 1-8. The right part of the area of interest is defined as scenario 2, which includes Zones 9-14. For performance comparisons, we include five individual predictors: Classification and Regression Tree (CART), Support Vector Machine (SVM), Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), Naive Bayes (NB), and one traditional ensemble predictor of Soft Voting (SV). When comparing machine learning prediction performance, it is impossible to define a single metric that provides a fair comparison in all possible applications. We focus on the metrics of prediction accuracy, which refers to the percentages of correct zone level localization.

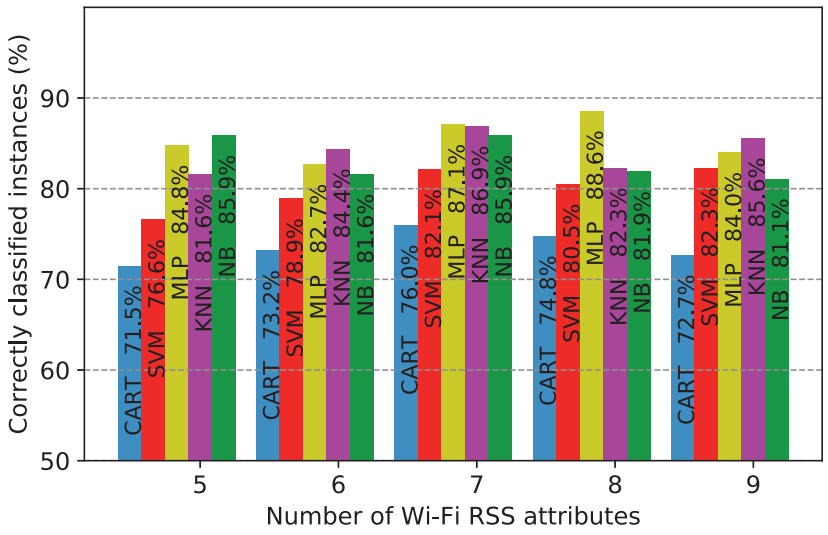


Figure 4.6: Individual predictors zone localization performance with different numbers of Wi-Fi RSS values in scenario 1.

Indoor Zone Level Localization Accuracy (COND)

First, we only use Wi-Fi RSS values as inputs features to machine learning algorithms in scenario 1. Figure 4.6 shows the classification accuracy of different individual predictors when different numbers of Wi-Fi RSS values are used. As we can see, starting from 5 RSS values, more RSS inputs increase the prediction accuracy for most of the predictors. Nevertheless, after 7 Wi-Fi RSS values are used, the improvement of adding more RSS values is almost negligible in almost all individual tested classifiers, and some of the predictors even got reduced accuracy when additional RSS values are considered. We think that the signal interferences may be the reason for the worse performance when more than 7 Wi-Fi RSS values are utilized. Therefore, we take the 7 largest Wi-Fi RSS values as the default configuration for the following experiments.

Next, we compare the classification accuracy when using only Wi-Fi RSS, Wi-Fi RSS plus MF, and Wi-Fi RSS plus MF and illuminance levels. Figure 4.7 shows the performance evaluation of the selected individual classifiers obtained with different feature combinations in scenario 1. The best performance is reached by the NB predictor, which achieves 90.3% of instances correctly classified when the fingerprint is composed by Wi-Fi RSS, MF readings, and illuminance levels.

Hyperparameters have significant impact on the performance of the learning-based

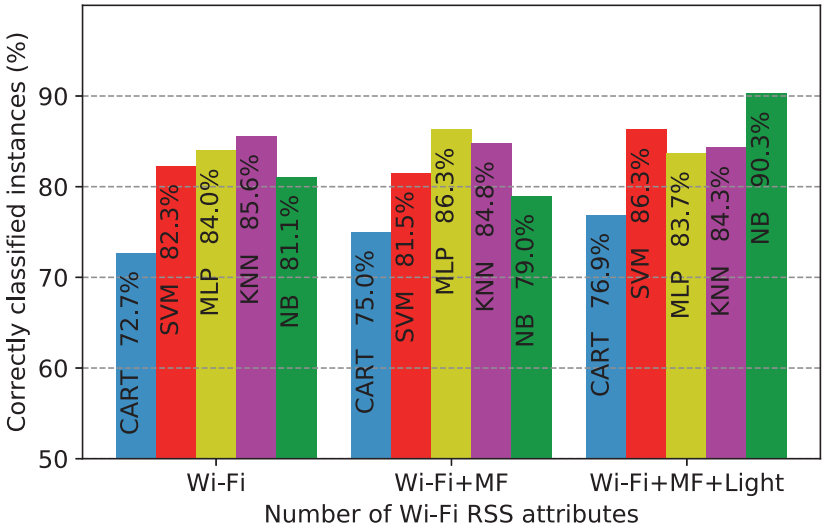


Figure 4.7: Individual predictors landmark prediction performance when using different features in scenario 1.

algorithm. Figure 4.8 shows the performance prediction of the selected individual and ensemble classifiers in scenario 1. Figure 4.9 presents the performance prediction of the selected individual and ensemble classifiers in scenario 2. The individual classifiers were set up with the hyperparameters optimized. The ensemble predictors use the outcomes of these individual classifiers as inputs. The classifiers are all fed with Wi-Fi RSS plus MF and illuminance levels. All the individual classifiers have improved performance, and NB even reaches an accuracy of 90.3%. Soft Voting uses the average predicted probabilities of CART, MLP, NB, KNN, and SVM to predict the room. SV can reach an accuracy of 87.7% in scenario 1 and 96.1% in scenario 2. Although all the tested traditional classifiers (i.e., CART, MLP, NB, KNN, SVM, and Voting) show high prediction accuracy, our proposed COND method outperforms them in both scenarios. COND achieves a prediction accuracy of 96.6% in scenario 1 and 96.8% in scenario 2.

In indoor environments, Wi-Fi RSS and MF measurements vary according to locations. However, these values will remain similar on nearby positions. For example, on locations close to zone borders, high similarities will be observed on the RSS values. These similarities could lead to misclassification problems. From Figure 4.7 we can see that KNN and MLP have better accuracy when both Wi-Fi RSS and MF readings are used. This is because KNN is an instance-based predictor, which uses entropy as

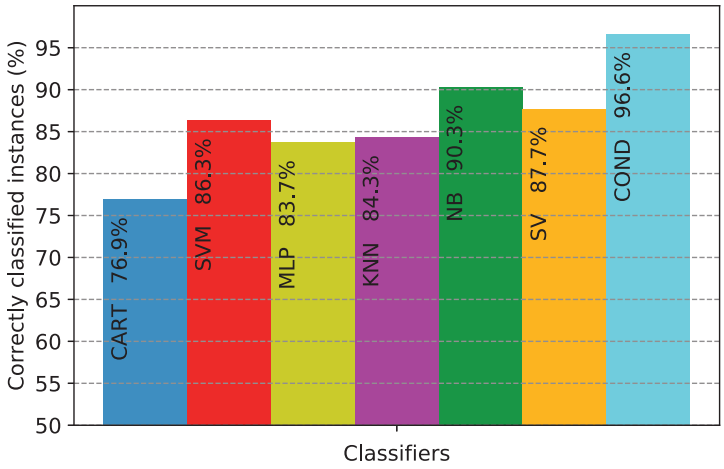


Figure 4.8: Zone level localization performance of individual predictors with optimized hyperparameters and ensemble predictors in Scenario 1.

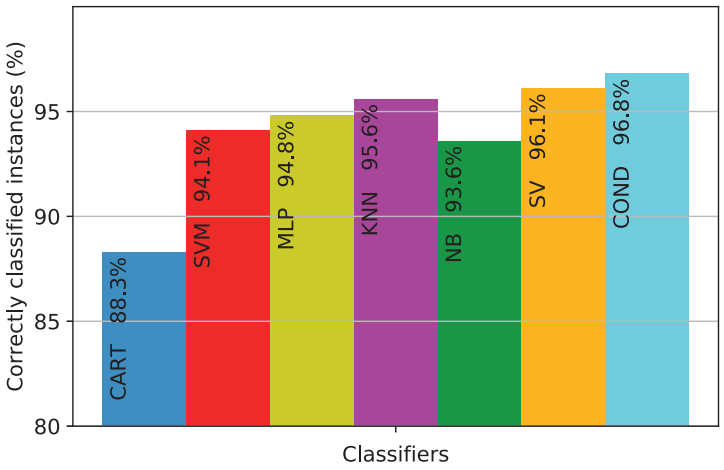
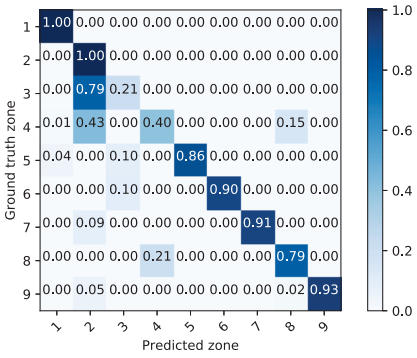


Figure 4.9: Zone level localization performance of individual predictors with optimized hyperparameters and Ensemble predictors in Scenario 2.

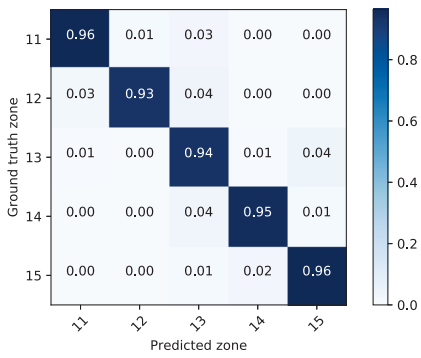
a distance measure to determine how similar two instances are. Thus, this method is more sensitive to small variations upon the instance as unity. Since CART is a decision tree machine learning algorithm, it builds the classification model by parsing the entropy of information on attribute level. It means that CART measures entropy in the attribute domain to decide which attribute should be included. Thus, the classification model is prone to misclassification in this room prediction problem. When the illuminance level is included as input feature to predictors, Naive Bayes outperforms others. This is because the feature of illuminance level is completely independent of other radio signal measurements, which fits with Naive Bayes' strong assumptions about the independence of each input variable.

To further explain how the individual and ensemble predictors perform in scenarios 1 and 2, we show the confusion matrix of room recognition using MLP, Naive Bayes (NB), KNN, SV and COND in Figures 4.10 and 4.11. We can observe that zone 3 is identified with accuracies of 21% by MLP, 100% by NB, 18% by KNN, 39% by Voting and 98% by COND. As a consequence, NB seems to be better in predicting zone 3 as compared to other individual and ensemble predictors. However, considering zone 4, NB achieves only 48% of accuracy, whereas KNN achieves 66%. We can see that the ensemble predictors adopt behaviors of different individual predictors. For instance, they adopt the good behavior of MLP and Naive Bayes, which leads to a much better prediction accuracy for zone 2. Unfortunately, ensemble predictors still have problems in some zones. For instance, SV achieves only 39% of instances correctly classified in zone 3. This is because most of the individual predictors that contribute to SV achieve low performance in that zone. It can be observed that in scenario 2 the SV ensemble predictor improves the accuracy compared to its constituent individual predictors. However, in scenario 1 the SV predictor does not achieve better performance than all its constituent individual predictors. It is because the prediction process of SV can be strongly influenced by individual predictors with low prediction performance.

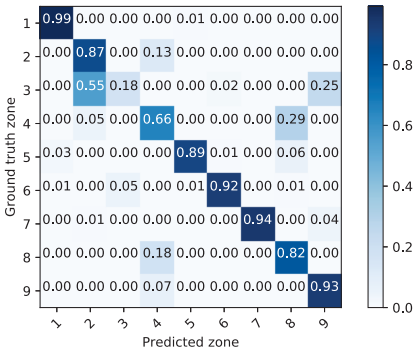
Our proposed COND method is able to overcome SV and all the individual tested predictors in both tested scenarios. Although COND and SV have the same constituent individual predictors, COND is able to perform better than SV. As it can be seen in Figure 4.11, COND overcomes SV in all zones. For instance, in zone 4, SV achieves 63% and COND 96% of correctly classified instances. This is because COND is able to balance out strengths and weakness of its constituent algorithms. This balance is made based on the observed prediction performance of each constituent classifier. Thus, we prove that COND is able to predict zones more reliably than the other tested



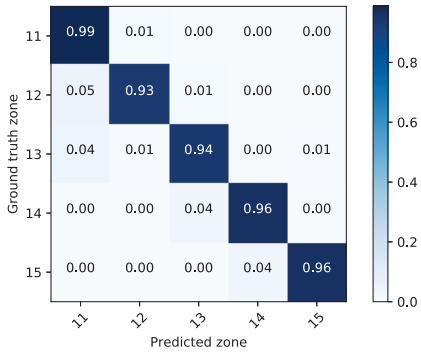
(a) MLP performance scenario 1.



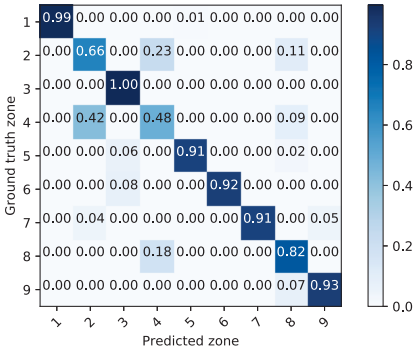
(b) MLP performacne scenario 2.



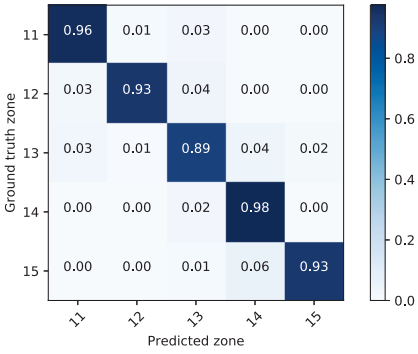
(c) KNN performance scenario 1.



(d) KNN performance scenario 2.



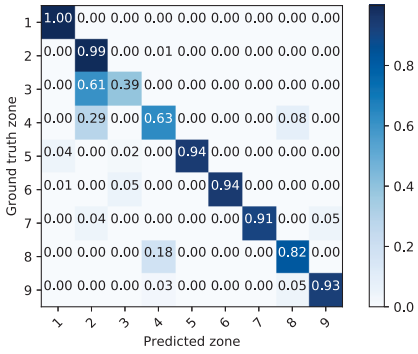
(e) NB performance scenario 1.



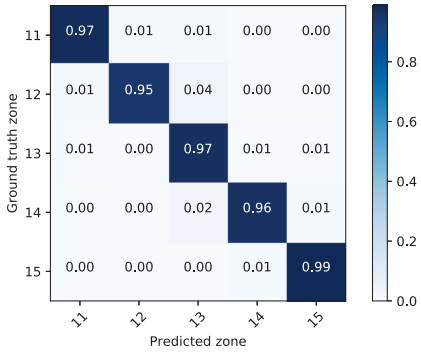
(f) NB performance scenario 2.

Figure 4.10: Individual predictors normalized confusion matrix in scenario 1 and 2.

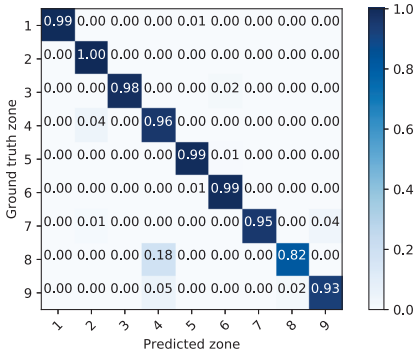
methods. Our prediction model allows the production of better zone prediction performance compared to individual and ensemble voting models.



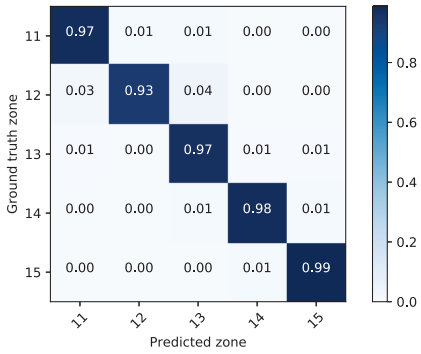
(a) SV performance scenario 1.



(b) SV performance scenario 2.



(c) COND performance scenario 1.



(d) COND performance scenario 2.

Figure 4.11: Ensemble predictors normalized confusion matrix in scenario 1 and 2.

Table 4.1: Theoretical Prediction Computational Complexity Analysis

Classifier	Time Complexity	Number of Operations
CART	$O(m_{features})$	10
MLP	$O(m_{features} \cdot Nn_1 + Nn_1 \cdot Nn_2)$	$10 \cdot 10 + 10 \cdot 14 = 240$
SVM	$O(m_{features} \cdot n_{sv})$	$10 \cdot 646 = 6460$
KNN	$O(n_{samples} \cdot m_{features})$	$17569 \cdot 10 = 175690$
NB	$O(m_{features})$	10
SV	CART+MLP+SVM+KNN+NB+$O(2 \cdot n_p \cdot n_c)$	$182410 + 140) = 182550$
COND	CART+MLP+SVM+KNN+NB+$O(n_p)$	$182410 + 5 = 182415$

COND Method Prediction Time Complexity Analysis

In this subsection, we present the prediction time complexity of the tested machine learning algorithms from a theoretical point of view. In machine learning, model complexity often depends on the number of extracted features and number of samples in the training database. Table 4.1 summarizes the prediction time complexity for KNN, MLP, SVM, NB, CART, COND, and SV considering the *Big O* notation. Moreover, Table 4.1 and Figure 4.12 show the theoretical number of operations considering 10 extracted features (i.e., 7 Wi-Fi RSS + MF + light) and the implementation setup described in subsections 4.5.1 and 4.6.1.

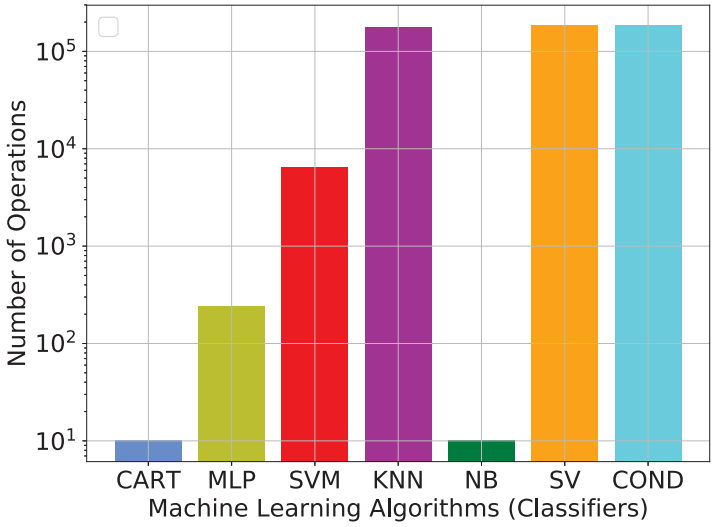


Figure 4.12: Theoretical Prediction Time Complexity

The ensemble predictor COND was applied to predict zones. We combine five individual machine learning algorithms in an ensemble model: KNN, MLP, SVM, NB, and CART. Therefore, the prediction time complexity of COND is the sum of prediction time complexities of five individual predictors plus the time complexity of the conditional probability computation. However, in ensemble learning, execution complexity time of meta-learners (i.e., conditional probability computation) is negligible and they have not much impact on execution time of base classifiers. CART prediction time complexity depends on the number of extracted features ($m_{features}$). In CART, each feature is analyzed in a decision node of the tree. Thus, the prediction time complexity of CART can be expressed by $O(m_{features})$. In MLP, the prediction time complexity depends on $O(m_{features})$, number of layers, and number of neurons in each layer (Nn). Thus, the prediction time complexity of MLP can be computed as $O(m_{features} \cdot Nn_1 + Nn_1 \cdot Nn_2 + ...)$, where Nn_i is the number of neurons in the i-th layer of the MLP model. In SVM, the prediction time complexity depends on the number of support vectors n_{sv} and number of features $m_{features}$. The prediction time complexity of SVM can be computed as $O(m_{features} \cdot n_{sv})$. KNN, as an instance-based classifier, has high prediction time complexity because instance-based classifiers essentially work by memorizing the training data. Thus, the prediction time complexity of KNN is $O(n_{samples} \cdot m_{features})$. NB computes the frequency of every feature value. Thus, the prediction time complexity of NB is $O(m_{features})$. The time complexity for computing the conditional probability is $O(n_p)$, where n_p is number of individual predictors, which are combined in the ensemble predictor COND. Therefore, the prediction time complexity of COND can be expressed as $O(m_{features} + m_{features} \cdot Nn_1 + Nn_1 \cdot Nn_2 + m_{features} \cdot n_{sv} + n_{samples} \cdot m_{features} + n_p)$. The prediction time complexity of SV is the sum of prediction time complexities of five individual predictors plus the time complexity of the soft voting rule computation. Thus, the prediction time complexity of SV can be expressed as $O(m_{features} + m_{features} \cdot Nn_1 + Nn_1 \cdot Nn_2 + m_{features} \cdot n_{sv} + n_{samples} \cdot m_{features} + 2 \cdot (n_p \cdot n_c)$, where n_c is the number of class labels.

Even though, the prediction performance of COND overcomes KNN, MLP, SVM, NB, CART and SV, the prediction time complexity of COND is higher than its constituent individual machine learning methods. When using 17569 training samples and 10 extracted features, the prediction time complexity is $O(18x10^4)$ (see Figure 4.12). However, as shown in Table 4.1, the prediction time complexity of the proposed ensemble learning method grows exponentially with the number of extracted features

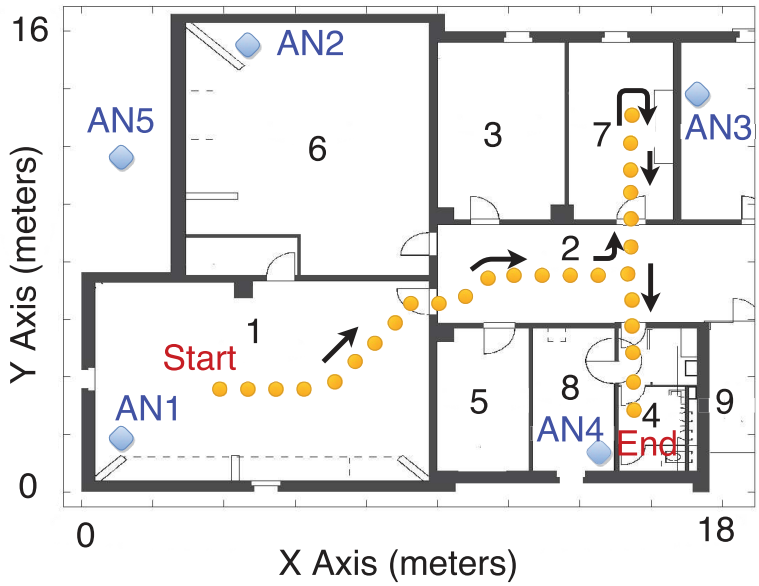


Figure 4.13: The First Trajectory, Zones, and ANs distribution (Diamond points: Anchor Nodes; Yellow points: trajectory)

and the number of training samples. This exponentially growing will lead to increase execution time in larger scenarios with bigger training databases (i.e., number of training samples and extracted features). Therefore, suitable processing resources to handle the algorithmic complexity of the ensemble predictor are the essential requirement to assure low prediction execution time.

4.6.2 Performance Evaluation HMM-d Method

Experiments were conducted in the building of the Institute of Computer Science at the University of Bern. A part of the third floor with an area of $288m^2$ ($18m$ x $16m$) was chosen to deploy the localization system. The smartphone is held by a person moving along three different trajectories (Figures 4.13, 4.14, 4.15). The zone detection method is launched every time a new fingerprint measurement is available (i.e., approximately twice per second). We define 8 zones in our environment (i.e., 8 class labels). Each zone is a wall separated area (i.e., rooms, corridor).

Figures 4.13, 4.14, and 4.15 present the physical distribution of zones, ANs, and trajectories. Additionally, to compare HMM-d to another ensemble learning model, we implemented a soft voting-based method. Hereafter, we refer to soft voting-based method as Voting method. The Voting method uses predicted zone labels from KStar,

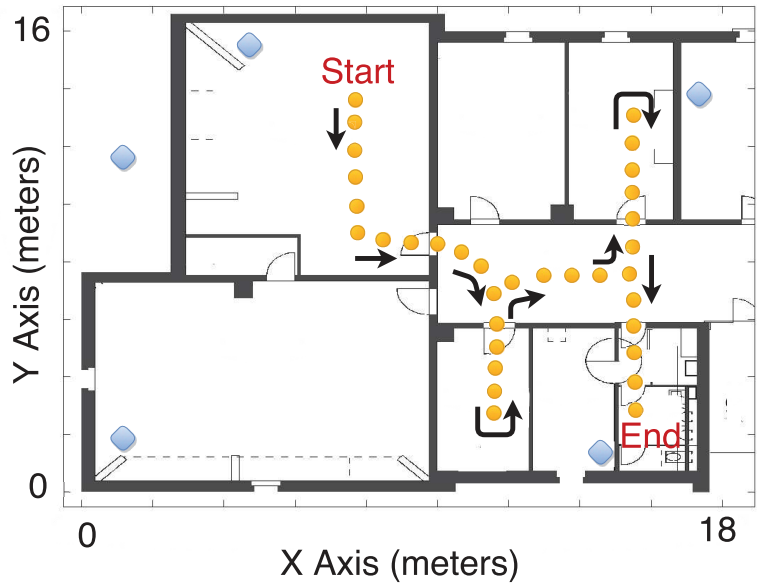


Figure 4.14: The Second Trajectory, Zones, and ANs distribution (Diamond points: Anchor Nodes; Yellow points: trajectory)

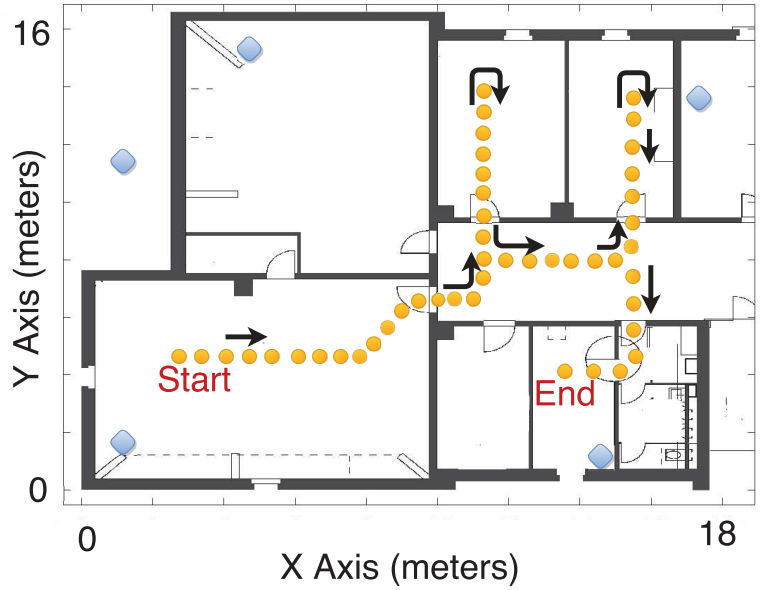


Figure 4.15: The Third Trajectory, Zones, and ANs distribution (Diamond points: Anchor Nodes; Yellow points: trajectory)

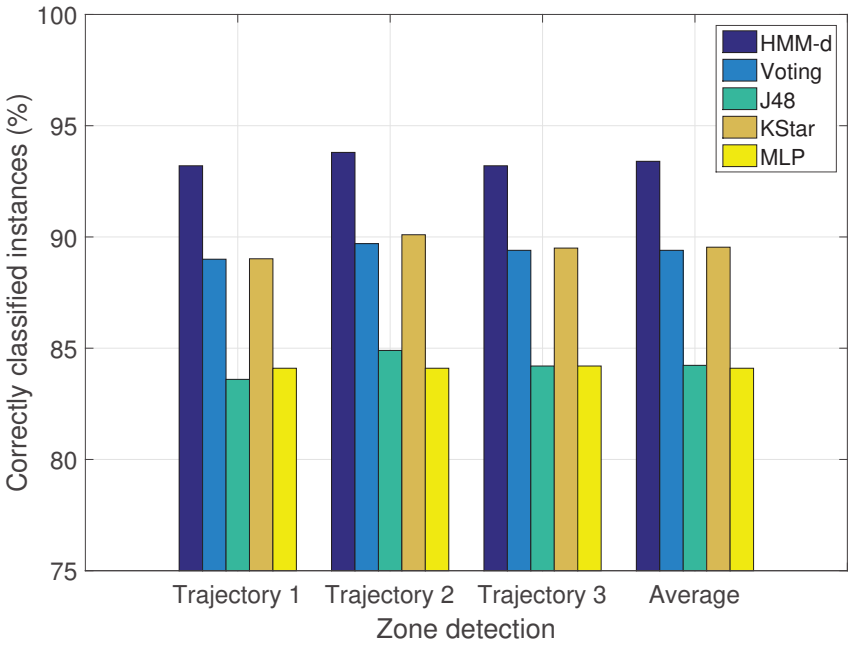


Figure 4.16: Predictive Model Accuracy

J48, and MLP for the soft voting rule. Further details about soft voting-based methods can be found in [91]. All the algorithms use the same fingerprint database of Wi-Fi RSS and MF readings, which have been measured during the data collection procedure. The individual predictors can be regarded as traditional fingerprint-based approaches, while the proposed HMM-d is a new ensemble predictor.

Zone Level Localization Performance (HMM-d)

To evaluate our prediction models, we consider three measures: prediction accuracy, F1 score, and processing time. In classification problems, accuracy is the ratio of correctly predicted observation to the total observations. F1 is the harmonic mean of precision and sensitivity. Precision is defined as the number of true positives (TP) divided by TP and the number of false positives (FP): TP/(TP+FP). Sensitivity is defined as the number of TP divided by TP and the number of false negatives (FN): TP/(TP+FN) [101]. Thus, F1 considers both performance measures, precision, and sensitivity. F1 can be written as follows:

$$F1 = 2 \cdot \frac{sensitivity \cdot precision}{sensitivity + precision}, \tag{4.10}$$

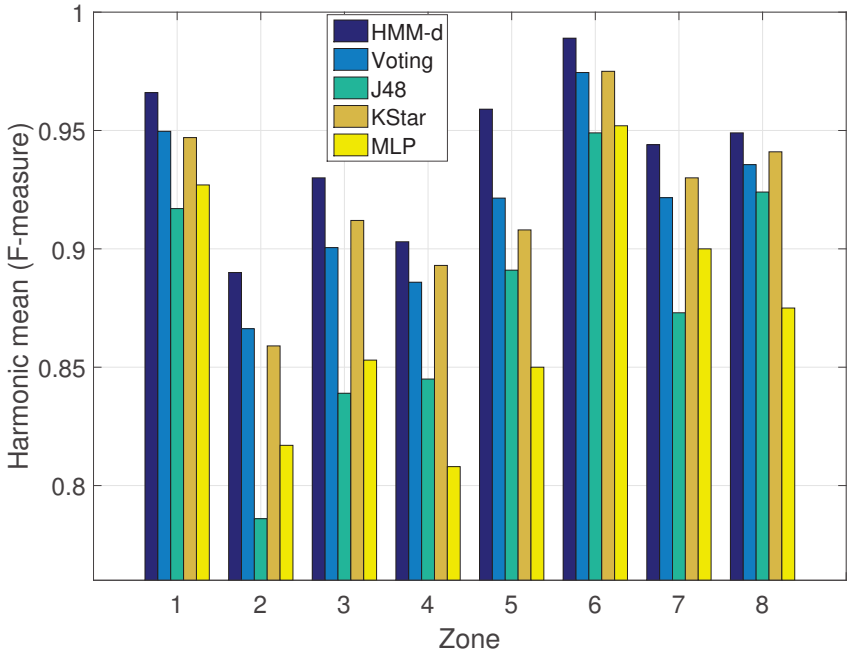


Figure 4.17: Zone Detection Performance F1 score

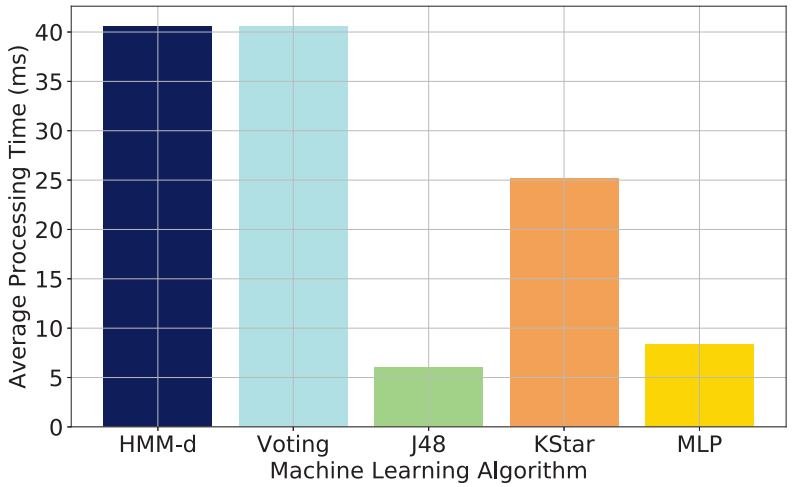


Figure 4.18: Average Processing Time of Ensemble and Individual Machine Learning Algorithms.

Figure 4.16 shows the accuracy of zone prediction for the five predictors in the three trajectories. Figure 4.17 presents the F1 score by zone for the learning algorithms. Figure 4.18 shows the average of prediction processing time for both ensemble models HMM-d and Voting and the individual machine learning methods (i.e., J48, KStar, MLP). Due to the hyperparameter optimization process, performance accuracy of the individual learning models (i.e., KStar, J48, and MLP) is higher than 80%. However, results show a clear improvement between the individual learning models and our ensemble learning model HMM-d. As it can be seen in Figure 4.16, our proposed HMM-d model outperforms KStar, J48, and MLP algorithms in the three tested trajectories. Accuracy of HMM-d is improved by 9.17%, 4%, 9.3%, and 9.2% compared to J48, KStar, MLP, and the Voting method respectively. Unlike Voting, HMM-d is able to combine zone transition information with individual learning methods to improve prediction accuracy.

Considering the F1 score, HMM-d outperforms others in all tested zones (i.e., classes). Therefore, HMM-d outperforms Voting, KStar, J48, and MLP considering accuracy and robustness. As it can be seen in Figure 4.16, accuracy of the learning algorithms remains similar (i.e., no significant difference) in all tested trajectories. However, if we consider sensitivity and precision as performance measure, it is clear to notice that zone 2 is the hardest zone to classify correctly (as shown in Figure 4.16). This result is explained as zone 2 corresponds to the corridor. Thus, fingerprints obtained in this zone are very similar to fingerprints obtained in adjacent zones, especially in the border areas. However, HMM-d improves the classification accuracy in this zone by 10.2%, 10.4%, 4% and 8% compared to Voting, J48, KStar, and MLP respectively. Thus, by combining HMM, KStar, J48, and MLP, our approach allows the production of better predictive performance compared to individual and ensemble voting models. Unlike Voting, the HMM-d model is able to achieve better prediction performance than individual learning methods in all zones.

Time Complexity Analysis HMM-d

We compare the prediction time complexity of the tested machine learning algorithms from a theoretical perspective. Table 4.2 summarizes the prediction time complexity of KStar, MLP, J48, Voting, and HMM-d considering the *Big O* notation. Additionally, Table 4.2 and Figure 4.19 show the number of operations considering 7 extracted features (i.e., 5 Wi-Fi RSS + MF) and the implementation setup described in subsection 4.5.1.

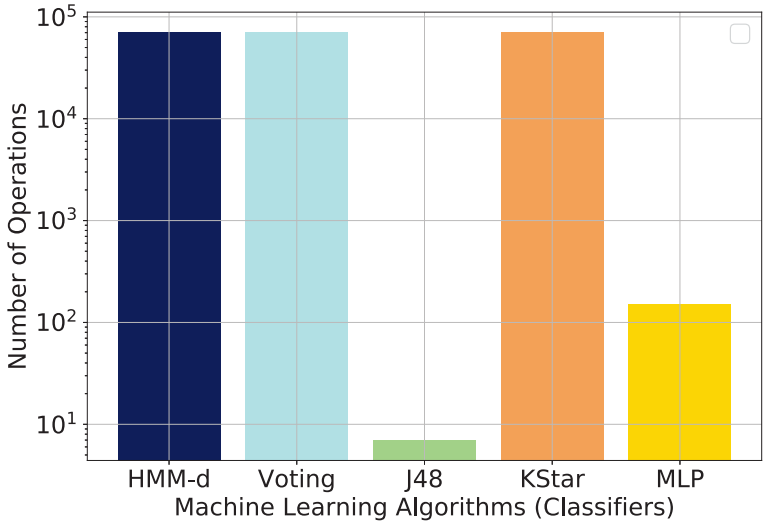


Figure 4.19: Theoretical Prediction Time Complexity

Table 4.2: Theoretical Prediction Computational Complexity Analysis

Classifier	**Time Complexity**	**Number of Operations**
J48	$O(m_{features})$	7
MLP	$O(m_{features} \cdot Nn_1 + Nn_1 \cdot Nn_2)$	$7 \cdot 10 + 10 \cdot 8 = 150$
KStar	$O(n_{samples} \cdot m_{features})$	$10000 \cdot 7 = 70000$
Voting	MLP+KStar+J48+$O(2 \cdot n_p \cdot n_c)$	$70157 + 2 \cdot 3 \cdot 8 = 70205$
HMM-d	MLP+KStar+J48+$O(n_s \cdot t)$	$70157 + 8 \cdot 3 = 70181$

The ensemble predictor HMM-d was applied to recognize the room. We combine three individual machine learning algorithms in a Hidden Markov Model: KStar, MLP, and J48. Therefore, the prediction time complexity of HMM-d is the sum of prediction time complexities of the three individual predictors plus the time complexity of the HMM. J48 prediction time complexity depends on the number of extracted features ($m_{features}$). In J48, each feature is analyzed in a decision node of the tree. Thus, the prediction time complexity of J48 can be expressed by $O(m_{features})$. KStar, as an instance-based classifier, has high prediction time complexity because instance-based classifiers essentially work by memorizing the training data. Thus, the prediction time complexity of KStar is $O(n_{samples} \cdot m_{features})$. The prediction time complexity of the MLP and Voting predictors are described in subsection 4.6.1. The prediction time complexity for HMM is $O(n_s \cdot t)$, where n_s is the number of states of the HMM and t is

the length of the observations sequence. Therefore, the prediction time complexity of HMM-d can be expressed as $O(m_{features} + m_{features} \cdot Nn_1 + Nn_1 \cdot Nn_2 + n_{samples} \cdot m_{features} + \cdot n_s \cdot t)$.

As shown in Table 4.2, the prediction time complexity of HMM-d is higher than its base predictors. When using 7 extracted features and 10000 training samples, the prediction time complexity of HMM-d is $O(70x10^3)$ (see Figure 4.19). However, the prediction time complexity of the HMM-d method grows exponentially with the number of extracted features and the number of training samples. This exponentially growing will lead to increase execution time in larger scenarios with bigger training databases (i.e., number of training samples and extracted features). Thus, appropriate processing resources to run ensemble machine learning models are the underlying requirement to guarantee low prediction execution time.

As shown in Figure 4.18, the average time of prediction (average time to perform a room prediction) is very similar in both HMM-d and Voting ensemble methods. However, HMM-d reduces the prediction processing time by $0.0125ms$ compared to the Voting method. This means that considering transition zone information in the prediction process takes lower computational efforts than processing the voting rule [91] that is used in voting methods. As expected, the processing time of the individual learning methods is lower than the ensemble learning methods. It is because the algorithmic complexity of the ensemble learning methods is higher compared to individual methods. Although the algorithmic complexity of HMM-d is higher than the individual methods, HMM-d model overcomes Voting, KStar, J48, and MLP methods by accuracy and robustness. In the following paragraphs, we provide a theoretical analysis of the prediction time complexity of the tested individual and ensemble learning methods.

4.6.3 Performance Evaluation Failure Recovery Method

We measure the ability of the failure recovery method to recover the system from localization failures. In this experiment, we simulate a localization failure by setting up the initial position of the set of particles in $p_0 = (18.4, 4.7)$. It is important to notice that p_0 is outside of the floor plan. Afterward, we asked the pedestrian to start the positioning system standing in an arbitrary known position. We repeated the experiment in seven rooms. Positions were recorded each second. Figure 4.20 depicts the recovery method behavior in each tested room. The X axis shows the elapsed time

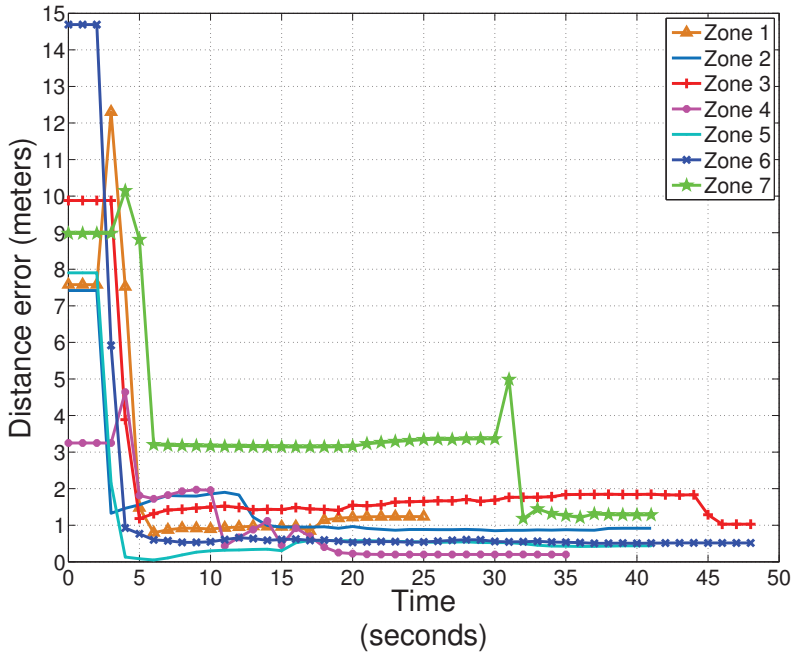


Figure 4.20: Recovering time from a localization failure.

until the convergence to a final position. The Y axis shows the positioning error.

Table 4.3: Localization Failure Recovery

Zone	Detection Time	Initial Pos. Error	Recovery Time	Final Pos. Error
1	3s	7.58m	20s	1.23m
2	3s	7.41m	40s	0.91m
3	4s	9.88m	47s	1.02m
4	4s	3.25m	20s	0.2m
5	3s	7.9m	33s	0.43m
6	3s	14.69m	43s	0.51m
7	4s	8.97m	37s	1.28m

Table 4.3 summarizes the failure detection time, initial positioning error, recovery time and final positioning error. As we can see from Table 4.3, the average time to detect the failure is 3.4s, and the average time to recover from localization error is 34.2s. This means that when the KRP happen during the indoor localization process, this can be automatically detected and recovered with acceptable time latency.

We define the localization error as the Euclidean distance between the ground truth

position and the final position reported by the system after recovering from localization failures. Thus, considering the localization accuracy, the mean error of the initial localization is nearly $8.5m$ if no failure recovery mechanism is applied. However, with our failure recovery mechanism, the mean error at the final position (after automatic recovery) is $0.8m$. The standard deviation is $0.41m$, which are much better than the results without using any recovery mechanisms. Based on these observations, we can highlight that despite the complex environment, the proposed failure recovery method achieves high accuracy and stable performance.

4.7 Conclusions

In this chapter, we presented a simple method to recover MCL based systems from localization failures. The proposed failure recovery method relies on machine learning techniques, which provide zone level localization. We evaluated our failure recovery method in a complex office-like scenario. Experiments showed that the failure recovery method is able to detect a localization failure in $3.4s$ of average. It achieved a mean error of $0.8m$.

As mentioned, the localization failure recovery method presented in this chapter relies on zone level localization results. Therefore, in this Chapter we also focused on providing enhanced machine learning methods for zone level localization. We proposed two ensemble machine learning methods (COND and HMM-d). The proposed methods integrate discriminative learning techniques in an ensemble learning method using ubiquitous ambient signal fingerprints. Thus, the proposed ensemble learning models achieve high prediction performance by combining less accurate individual discriminative learning models.

We validated the prediction performance of COND and its constituent machine learning methods by using different smartphone sensor measurements, such as Wi-Fi RSS, MF readings, and illuminance levels. Moreover, we compared the prediction performance of COND to SV method, which is a traditional ensemble learning method. Evaluation results show that COND achieves the best indoor landmark localization accuracy of 96.8% compared to its constituent machine learning methods and SV. Moreover, we compared the prediction time complexity of the tested machine learning algorithms. Although, the prediction performance of COND overcomes KNN, MLP, SVM, NB, CART and SV, the prediction time complexity of COND grows exponentially

considering *big O* notation.

The HMM-d method adopts Hidden Markov Models to integrate information about transition probabilities between zones and discriminative learning methods. Thus, unlike traditional machine learning models, HMM-d integrates observed fingerprints and coarse-grained floor plan information to predict class levels. We evaluated HMM-d method in a complex real-world indoor environment. Evaluation results indicate that our proposed approach is more accurate and robust than its constituent individual machine learning methods and SV. Moreover, we compared the prediction time complexity of HMM-d and its constituent machine learning algorithms. Although, the prediction performance of HMM-d overcomes KStar, MLP, J48, and Voting, the prediction time complexity of HMM-d grows exponentially considering *big O* notation. This exponentially growing will lead to increase execution time in larger scenarios with bigger training databases (i.e., number of training samples and extracted features). Thus, appropriate processing resources to run ensemble machine learning models are the underlying requirement to guarantee low prediction execution time.

Although the growing development in mobile devices such as smartphones, mobile devices still need to deal with limited processing and power resources. In this chapter, we have shown that our proposed localization approaches achieve accurate localization performance. However, time complexity grows exponentially with the complexity of the localization method. It is clear that the algorithmic complexity of the localization methods is constrained by the limited computation and power resources on the mobile device. To solve this problem, in the following Chapters we focus on MEC-based infrastructure to provide robust indoor localization and tracking for wireless mobile devices. MEC-based systems enable real-time functionalities to the system.

Part II

MEC Architecture for Indoor Tracking

Despite the fast development and new advances in mobile devices (e.g., smartphones, etc.), mobile devices still need to deal with limited battery and limited processing resources, especially for interactive resource-intensive mobile applications such as machine learning, and real-time localization. Although, CC enables high storage capacity and high computational processing power, network and computational overhead at the central cloud increases. This leads to performance issues for real-time applications, such as indoor positioning, where low latency is a fundamental requisite. As an alternative, MEC has been introduced to minimize the network overhead while still providing CC functionalities. The MEC architecture brings processing capabilities in close proximity to the user. This improves latency and enables real-time functionalities. Moreover, the system performance is not affected by the limited battery and processing resources of mobile devices [75] [105]. In Part II, we propose MEC architectures where lightweight algorithms run on the mobile target devices, whereas heavy calculations are offloaded to nearby edge servers. Thus, algorithmic complexity is not constrained by the limited computational resources of target mobiles devices. Chapter 5 presents a two layers MEC-based localization and tracking system for smartphones. The proposed tracking system includes a particle filter-based reinforcement learning approach for reliable indoor localization. In Chapter 6, we extend our work to provide indoor localization to IoT devices. Thus, Chapter 6 presents a three layers MEC-based tracking system. The proposed system includes a cloud layer to provide centralized services, such as remote monitoring and storage of historical localization information of IoT devices.

5

MEC based Reinforcement Learning Method for Indoor Tracking with Failure Recovery

5.1 Introduction

In this chapter, we exploit MEC architecture to extend our localization approaches, which were introduced in Chapter 3 and 4. We propose a distributed architecture for indoor tracking, where lightweight methods run on the mobile target device (i.e., smartphone) and heavy computations are offloaded to nearby edge servers. The tracking approach is based on particle filter and reinforcement learning methods to guarantee system robustness against localization failures such as global localization and KRP. Moreover, we exploit zone level localization to choose the proper ranging models that are specific for each zone. If the mobile target device (i.e., object to be located) and a ranging Anchor Node (AN) are at the same zone, the system adopts LOS ranging models with respect to this AN. Furthermore, ranges to ANs located at

different zones than the mobile target are calculated by NLOS ranging models.

Further, we include reinforcement learning techniques into the resampling method to assure the placement of samples (i.e., particles) over areas where the desired distribution is large (i.e., areas with high probability of containing the ground truth position). This scheme reduces convergence time and provides autonomy and robustness to the system against localization failures.

Figure 5.1 shows the architecture of our proposed system, which is a two-tier architecture supporting distributed machine learning operations. The main contributions of this Chapter are as follows.

- We propose a MEC architecture for indoor tracking. We propose a distributed machine learning-based network architecture, where lightweight algorithms are running on the mobile target devices with limit resources and heavy ML calculations (proposed PFRL algorithm) are offloaded to nearby edge servers to support complex and heavy calculations. Compare to terminal-based indoor positioning system, a MEC-based system is able to run algorithms with high complexity.
- We design a particle filter-based reinforcement learning (PFRL) algorithm for reliable wireless indoor positioning. The particle filter fuses the predicted zone, radio-based ranges, IMUs, and coarse-grained floor plan information to achieve accurate and stable real-time indoor tracking performance. In the particle filter, we provide a reinforcement learning-based resampling method to guarantee system robustness against localization failures. To deal with multi-path effects, we exploit zone level localization to choose the proper ranging models that are specific for each zone. If the mobile target (i.e., object to be located) and a ranging Anchor Node (AN) are at the same zone, the system adopts LOS ranging models with respect to this AN.

We evaluate our system in complex office and classroom-like environments along five different moving paths. Our proposed tracking approach can achieve 0.97 meters for mean localization error and recovery time latency of 1.5 seconds, which is more accurate and stable than terminal-based localization methods.

The rest of the Chapter is organized as follows. Section 5.2 describes the localization system. Implementation details are presented in Section 5.3. Section 5.4 describes

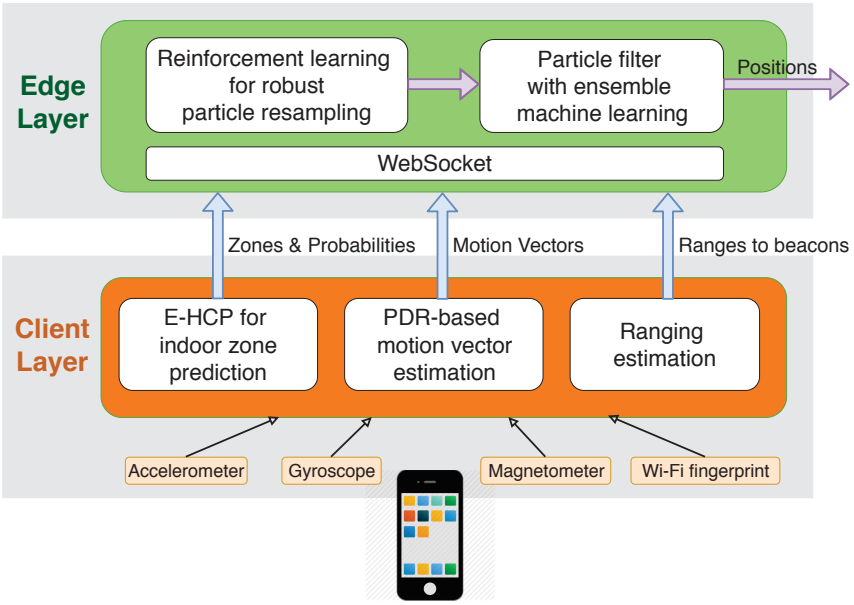


Figure 5.1: Distributed Machine Learning System Architecture for Reliable Wireless Indoor Positioning.

the localization performance evaluation of our approach. Section 5.5 concludes the Chapter.

5.2 System Overview

This section presents the design details of the proposed particle filter-based reinforcement learning approach for wireless indoor positioning. Figure 5.1 summarizes the architecture of our proposed approach, which includes two layers: a Client layer and an Edge layer. The Client layer includes mobile clients to be located, such as smartphones, tablets, etc. The Edge layer includes edge servers, which are responsible for hosting complex positioning algorithms. Due to limited amount of resources available, mobile clients host components that are able to process low computation overheads, which include: a HMM-based ensemble predictor for indoor zone prediction, an enhanced ranging model, a PDR-based movement detection method and a floor plan component that defines a discrete system state, the map likelihood (i.e., allowed areas to move), and the transition model (i.e., physical distribution of zones). Edge servers host the proposed PFRL algorithm, which considers the outputs of mobile clients as inputs to estimate the real-time positions of mobile devices. PFRL includes two

parts: a particle filter-based ensemble predictor to provide high positioning accuracy and a reinforcement learning approach, which builds on top of the particle filter to guarantee system robustness against failures. Details of each component and the interconnections between client and edge layers are described in the next subsections.

5.2.1 Mobile Client - Ensemble HMM-Conditional Performance Learning (HMM-d) for Indoor Zone Prediction

We propose a Hidden Markov-based ensemble learning method (HMM-d) for zone prediction. The key idea of HMM-d is to combine conceptually different individual machine learning models in an HMM. Thus, we combine some individual machine learning algorithms to improve prediction performance compared to individual models. Design and implementation details about the HMM-d method can be found in Chapter 4.

In any model with hidden variables such as HMM-d, a decoding task is to determine the sequences of variables that is the underlying source of an observation sequence. The decoding task can be described as the process to solve the following equation:

$$p(z_{l_t} \mid q_{l_t}) = \frac{p(q_{l_t} \mid z_{l_t}) \cdot p(z_{l_t})}{p(q_{l_t})} \tag{5.1}$$

where $P(z_{l_t} \mid q_{l_t})$ is the probability being located at zone z_{l_t} given the observation q_{l_t} at time t. Therefore, considering a sequence of observations $q_{0_0}, ..., q_{l_t}$, and an HMM model $\lambda = \{\pi, A, B\}$, Equation 5.1 can be computed by applying the Viterbi algorithm [39].

The zone prediction method provides three functionalities to the whole localization approach: First, this method provides zone-level localization. If the target object and ranging Anchor Node (AN) are at the same zone, the system adopts Line of Sight (LOS) ranging models with respect to this AN. Furthermore, ranges to ANs located at other zones than the target object are calculated by Non Line of Sight (NLOS) ranging models. Second, the zone prediction method supports the robustness against localization failures. This method provides information about how to distribute particles in the resampling process to avoid and to recover the system from localization failures. Thus, particles are distributed only at zones with higher probability of being the actual zone where the target is located. This avoids localization failures and provides faster

particle convergence. Third, the zone prediction method supports system recovery from localization failures. Thus, after a localization failure, the particle distribution density will be higher at zones with higher probabilities of being the zone where the target is located. This reduces the time needed for recovering the system from localization failures.

5.2.2 Mobile Client - PDR-based Motion Vector Estimation

PDR methods estimate the displacement of the pedestrian by detecting changes in a previously estimated position. The pedestrian displacement is estimated by using three device embedded sensors: accelerometer, geomagnetic field sensor, and gyroscope. At time t, the displacement of the pedestrian is defined by the motion vector $M_t = [\ell_t, \theta_t]$. Thus, M_t is passed to the Particle Filter component at instant t when a displacement (e.g., step) of the pedestrian is detected. Additional details about the PDR method can be found in Chapter 3.

5.2.3 Mobile Client - Ranging Estimation Process

Ranges can be derived by using signal parameters such as RSSI. In theory, RSSI monotonically decreases with increasing propagation distance [59]. However, in complex indoor environments, WiFi signals suffer from random variations. To reduce ranging errors introduced by NLOS and multi-path propagation, we propose a propagation model by combining Log Distance Path Loss (LDPL) [93] and a Nonlinear Regression Model (NLR) [59] [18]. Our propagation model can be written as follows:

$$r = \begin{cases} 10^{(\frac{Pw(r_0)-Pw(r)}{10\cdot\gamma})} & \textit{if Tx and Rx are at the same zone} \\ \alpha \cdot e^{(\beta \cdot Pw(r))} & \textit{if Tx and Rx are at different zones,} \end{cases} \tag{5.2}$$

where *Tx* and *Rx* are the transmitter and receiver devices respectively. Variable $P_w(r_0)$ refers to the power loss in a free space, $P_w(r)$ is the received signal power in a propagation distance r. Variable γ is the path loss efficient [93], and both α and β are environmental variables [59].

Algorithm 4 PF-MLM

Result: Mobile client position

Calculate the initial zone probability distribution: $p(z_{l_0} \mid q_{l_0}) = HMM\text{-}d()$

Distribute particles based on $p(z_{l_0} \mid q_{l_0})$

Initialize particle's weights: $W_0^i = 1/N_p$, $i = 1, 2, ..., N_p$

while *Localizing* **do**

Update the particles: $X_t^i = G \cdot X_{t-1}^i + \eta$

Calculate the ranging likelihood: $P(\hat{d}_{j,t} \mid X_t^i) = \frac{1}{\sigma_j \sqrt{2\pi}} \exp^{-\frac{[\hat{d}_{j,t} - \sqrt{(x^i - x_j)^2 + (y^i - y_j)^2}]^2}{2\sigma_j^2}}$

Calculate the zone probability distribution: $p(z_{l_t} \mid q_{l_t}) = HMM\text{-}d()$

Calculate the zone likelihood: $P(z_{l_t} \mid X_t^i) = \frac{P(X_t^i \mid \hat{z}_{l_t}) \cdot P(\hat{z}_{l_t})}{P(X_t^i)}$

Compute unnormalized weights: $\hat{w}_t^i = P(X_t^i \mid \hat{z}_{l_t}) \cdot \prod_{j=1}^{M} P(\hat{d}_{j,t} \mid X_t^i)$

Normalize weights: $w_t^i = \frac{\hat{w}_t^i}{\sum_{n=1}^{N} \hat{w}_n^i}$

Resample the particles

Calculate the degree of depletion Run RL method for robust tracking resampling

Compute the estimated position:$X_t = \sum_{i=1}^{N} w_t^i \cdot x_t^i$

end

5.2.4 Edge Server - Particle Filter with Ensemble Machine Learning

Particle filtering is able to solve estimation problems recursively as observations become available. The objective is to determine the posterior distributions of the system's states given some noisy observations. The posterior probability is expressed as a set of weighted samples (also called particles). Thus, the posterior probability distribution is computed based on some observation O_t at time t [20]. A time t, the particle system state vector X_t can be defined by Equation 2.39. At time t, the set of particles can be expressed as $P_t = [X_t^i, W_t^i], i = 1, ..., N_p$, where N_p is the number of particles, X_t^i is the state vector, and W_t^i is the associated weight of the i-th particle at time t.

State vector X_t^i of each particle is updated from the particles at the previous time interval X_{t-1}^i based on Equation (2.43). Thus, the new set of particles P_t is generated from P_{t-1}. Particles are not allowed to move through restricted areas, (e.g., movement through walls is not allowed).

Observation Model for Data Fusion

Particles are propagated based on Equation (2.43). Afterwards, the associated weight w_t^i of the propagated particles must be calculated. At time t, the associated weight $p(O_t \mid X_t^i)$ is calculated based on the likelihood of the observations conditioned on current particle state X_t^i. The observation vector O_t contains the ranging information to different ANs and the predicted zone information. Thus, at time t, the observation vector can be expressed as $O_t = [d_t, q_{l_t}]$, where d_t contains ranges to different ANs and $q_{l_t} \in C$ contains the predicted zone information. Ranges are computed by the Ranging Estimation process presented in Section 5.2.3. The zone prediction information is provided by the HMM-d method presented in Section 5.2.1.

Since the ranging method (i.e., the method to estimate ranges) and the HMM-d method for zone prediction are completely different, we can assume that $p(q_{l_t} \mid X_t^i)$ and $p(d_t \mid X_t^i)$ are independent of each other. Therefore, the probability $p(O_t \mid X_t^i)$ can be expressed as follows:

$$p(O_t \mid X_t^i) = p(d_t \mid X_t^i) \cdot p(q_{l_t} \mid X_t^i) \tag{5.3}$$

We refer to $p(d_t \mid X_t^i)$ as the ranging likelihood, and $p(q_{l_t} \mid X_t^i)$ as the zone likelihood.

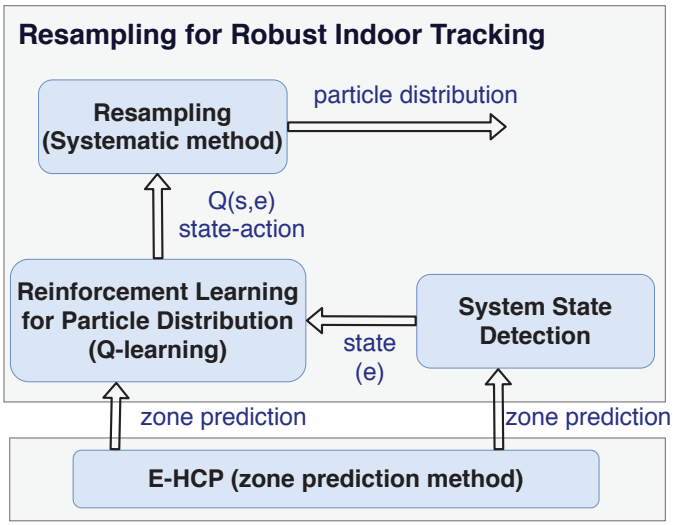


Figure 5.2: Reinforcement Learning Method for Robust Indoor Tracking Resampling.

Therefore, the associated weight $w_t^i = p(O_t \mid X_t^i)$ of each particle is given by the ranging and zone prediction information. The particles at the absolute position (x_t, y_t) with low probability to observe d_t^j in their position will be assigned a small ranging likelihood. Particles positioned in zones with low probability of observing q_{l_t} will be assigned small zone likelihood values.

The ranging likelihood $p(d_t \mid X_t^i)$ can be calculated by Equation 3.5.

Zone likelihood refers to the zone prediction information. Therefore, zone likelihood is the probability of observing q_{l_t} in the current particle state X_t^i. The value of the variable q_{l_t} is computed by the HMM-d method.

5.2.5 Edge Server - Reinforcement Learning for Robust Indoor Tracking Resampling

In particle filter localization approaches, the localization process performs poorly if the proposal particle distribution (i.e., distribution used to generated samples) places too few samples in areas where the desired posterior distribution is large. Such behaviour leads to increase convergence time of the particle filter. Moreover, an unsuitable proposal distribution could trigger localization failures such as the kidnapping robot problem. To mitigate these problems, we propose to use adaptive proposal distributions in addition to the resampling method. The proposal distributions are

built based on a reinforcement learning method, which relies on the HMM-d outcomes and the current state of the system. Figure 5.2 depicts the architecture of the proposed reinforcement learning method for robust indoor tracking. The proposal distribution assures the placement of samples over the areas where the desired distribution is large. The tracking algorithm learns by itself which proposal distribution suits better at each state of the system. This scheme reduces convergence time and provides autonomy and robustness to the system.

Resampling Method

Resampling is a fundamental process for particle filters. Without resampling, particle filters will produce a degenerate set of propagated particles (i.e., most of the particles with negligible weight). The resampling process modifies the weighted approximate density p to an unweighted density $\hat{p}$ by eliminating particles with low importance weights (i.e., small associated weight) by multiplying particles having high importance weights (i.e., high associated weight). The new density $\hat{p}$ is called the proposal distribution. Therefore, $p(X_t \mid q_{1:t}) = \sum_{i=0}^{N_s} w_t^i \delta(X_t - X_t^i)$ is replaced by $p(\hat{X}_t \mid q_{1:t}) = \sum_{i=0}^{N_s} \frac{n_i}{N_s} \delta(\hat{X}_t - \hat{X}_t^i)$, where n_i is the number of copies of particle X_t^i in the new set of particles $\hat{P}_t$. There are many methods to generate $\hat{P}_t$ [28]. We perform the resampling process by using the systematic method. The systematic resampling method aims to prevent the degeneracy of the propagated particles by modifying the set P_t to $\hat{P}_t$. Particles from P_t with higher weights are more likely to be included in the new set of particles $\hat{P}_t$. Thus, in the next iteration, more particles will be propagated in zones with large probability masses [56]. Before resampling, the weights W_t^k are normalized, i.e., $\sum_{k=1}^{N_p} W_t^k = 1$. Then, a set of N_p numbers u_t^n is generated from an uniform distribution. This set of numbers is used to select N_p particles from P_t. Thus, the particle x_t^n is selected in the n-th iteration if the following condition is satisfied:

$$S_t^{m-1} < u_t^n \leq S_t^m, m = 1, ..., N_p, \tag{5.4}$$

, where

$$S_t^m = \sum_{k=1}^{m} W_t^k, \tag{5.5}$$

The interval $(0,1]$ is divided into N_p disjoint sub-intervals $(0, 1/N_p] \cup ... \cup (1 - 1/N_p, 1]$. Then, u_t^1 is generated as a random number from the uniform distribution on $(0, 1/N_p]$.

Figure 5.3: PFRL transition model learner agent.

The remaining u_t^n numbers are obtained from u_t^1 as follows:

$$\begin{aligned} u_t^1 &\sim U(0, 1/N_p], \\ u_t^n &= u_t^1 + \frac{n-1}{N_p}, \quad n = 2, 3, ..., N_p, \end{aligned} \tag{5.6}$$

After generating the set of u_t^n numbers, the new set of particles $\hat{P}_t$ is generated by selecting N_p particles from P_t based on the condition presented in Equation 5.4. Although the systematic resampling method can achieve high performance, this method alone does not guarantee the system to avoid and recover from localization failures.

Reinforcement Learning Method for Particle Distribution

Considering a reinforcement learning context, our tracking algorithm is modeled as the learner agent (LA). The LA provides localization and at the same time learns the optimal behaviour to prevent and recover the system from localization failures. The Q-learning [109] approach is adopted as a reinforcement learning method. Q-learning provides agents with the ability to learn how to proceed optimally by experiencing the consequences of actions [109]. In Q-learning, the LA evaluates the consequences of an action at a particular state. This evaluation is performed in terms of an immediate penalty or reward. Thus, by trying all actions in all states repeatedly, LA learns the optimal behaviour at each state. Figure 5.3 shows the states and transition model of our proposed reinforcement learning model. Since the purpose of the LA is to provide autonomy and robustness to the system, we defined three states: Starting state, which is achieved when the system is started, Localizing state, which is achieved when the system is providing localization service, and Failing state, which is achieved when some localization failure is detected. Elements of the set of actions are as follows.

- Action e_1 defines a uniform particle distribution across the whole target area. This action can be performed in the Starting and Failing states.

- Action e_2 defines a particle distribution across the predicted zone. Zone information is computed by the HMM-d method. This action can be performed in the Starting and Failing states.
- Action e_3 defines a particle distribution based on the predicted zone probability distribution. Zone probability distribution is computed by the HMM-d method. This action can be performed in the Starting and Failing states.
- Action e_4 defines a particle distribution of the g% worst evaluated particles (i.e., particles with the lowest weight) across the predicted zone. This action can be performed in the Localizing and Failing states.
- Action e_5 defines a particle distribution of the g% worst evaluated particles based on the predicted zone probability distribution. This action can be performed in the Localizing and Failing states.

Q-learning algorithm performs the learning process based on the Bellman equation as follows:

$$Q(s,e) \leftarrow (1-\alpha) \cdot Q(s,e) + \alpha \cdot [R(s,e) + \gamma \cdot \max(Q(s',e'))], \tag{5.7}$$

where $Q(s,e)$ determines the quality of a state-action combination. Thus, when action e is performed in state s, $Q(s,e)$ is updated based on Equation 5.7. The learning rate is defined by α, which determines how valuable recent information is for the learning process. Thus, if $\alpha = 0$, the LA exploits only previous learned knowledge, while $\alpha = 1$ makes the LA to consider only the most recent information. The parameter γ defines the discount rate, which determines what percentage of a future reward must be considered in the training process. The variable $\max(Q(s',e'))$ is the maximum achievable $Q(s',e')$ value, which is possible to obtain in the next state s' by performing e'. The function $R(s,e)$ computes the reward of performing action e at state s. Further details about the Q-learning algorithm can be found in [71, 109].

We define $R(s,e)$ as a function to compute the degree of depletion in the particle filter method. The degree of depletion describes the rate of particles having a negligible weight. Some particles can be located away from the ground truth location. Therefore, these particles are evaluated with nearly negligible weights. The density of particles should be high in high-probability zones, and low in low-probability zones. The

effective number of samples (N_{eff}) is an indicator of the degree of depletion [65]. Therefore, N_{eff} measures how efficiently the particle distribution is representing the ground truth location. Since the degree of depletion indicates the quality of particle distributions, and the effective number of samples N_{eff} is an indicator of the degree of depletion, we define $R(s,e) = N_{\text{eff}}$. Thus, the value of N_{eff} for N_p number of particles can be calculated as follows:

$$R(s,e) = N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_p}(w^i)^2} \tag{5.8}$$

where w^i is the particle's associated weight.

System State Detection Method

Q-learning is a method that evaluates which action to perform based on the system state. Therefore, detecting the current system state is an essential requirement. The system moves from the Starting state to the Localizing state when the starting position is established (i.e., when particles converge to the initial starting position). Thus, the LA must perform the optimal action for fast and accurate convergence. The Starting state is clearly identifiable. When a localization failure is detected, the LA must perform the optimal action for a quick and accurate recovery.

To detect the current system state, we propose a novel and effective method, which is based on the zone prediction information provided by the HMM-d method, the current particle distribution and the current degree of depletion in the particle filter method. Therefore, after performing the systematic resampling process, the system state detection method is executed. Thus, if any particle is placed in the predicted zone, and the effective number of samples is lower than a predefined threshold T_h, the algorithm assumes a localization failure. Figure 5.4 shows the flowchart of the system state detection method.

5.2.6 Data flow between Mobile Client and Edge Server

Mobile clients collect raw data using on-board sensors, such as accelerometer, gyroscope, magnetometer, Wi-Fi signals, etc. Such data are processed on mobile devices using lightweight machine learning algorithms, such as HMM-d zone prediction (Section III.A). Afterwards, mobile clients could derive the zone information with

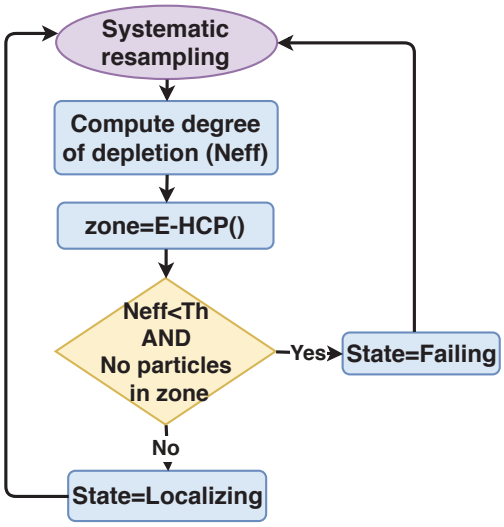


Figure 5.4: PFRL system state detection method flowchart.

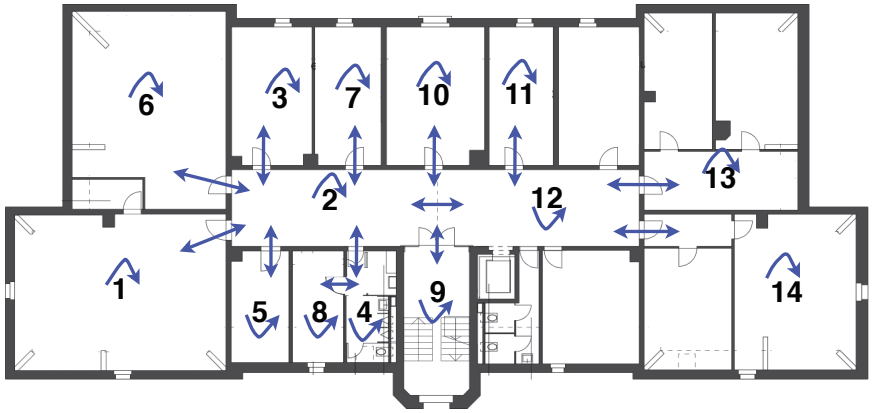


Figure 5.5: Physical layout, zone definition and graphical transition model in scenario 1

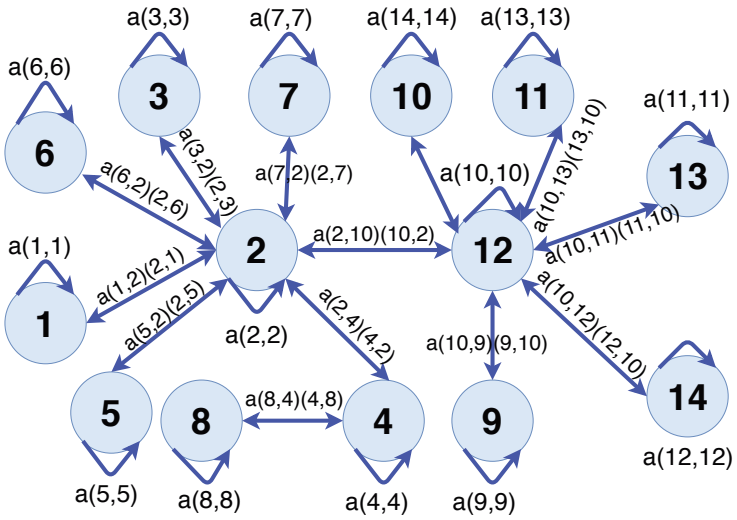


Figure 5.6: Scenario 1: Transition information among zones are used to define the transition model for HMM-d method.

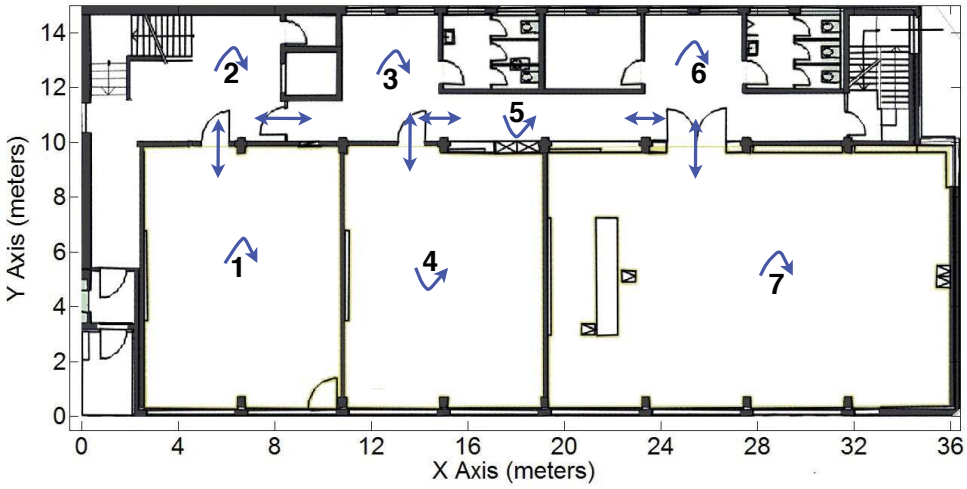


Figure 5.7: Physical layout, zone definition and graphical transition model in scenario 2

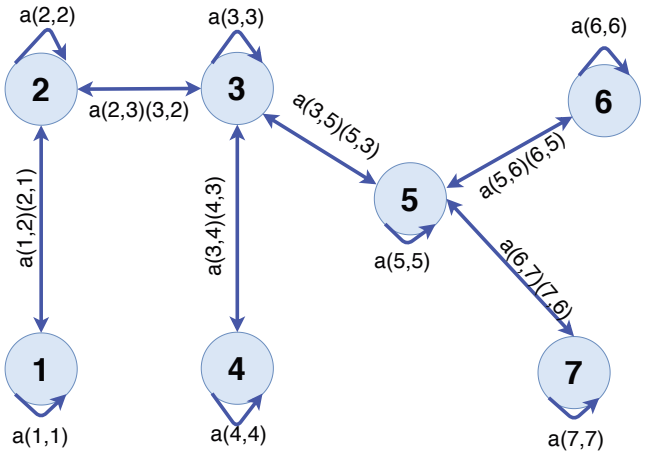


Figure 5.8: Scenario 2: Transition information among zones are used to define the transition model for HMM-d method.

probabilities, motion vectors, and ranges to relevant beacons. This information are then transmitted to the edge server using the WebSocket for further processing. At the edge server, heavy machine learning approaches, including ensemble learning and reinforcement learning, will be applied on the received information to estimate the accurate indoor positions.

5.3 Implementation

We implemented the proposed system on edge servers and smartphones. It comprises three main components: a mobile target (MT), a set of Wi-Fi Anchor Nodes (ANs), and an edge server. ANs are off-the-shelf Wi-Fi access points, which are placed at certain locations to guarantee the maximum coverage for the indoor areas. We used a Motorola Nexus 6 smartphone with 3 GB RAM and Quad-core 2.7 GHz CPU as experimental device. A HP EliteBook with 8 GB RAM and 2.30 GHz Intel Core i5-5300U processor is used as edge server.

The localization algorithm (i.e. Particle filter) and the reinforcement learning-based method (i.e., Q-learning) are implemented at the edge server by using Python 2.7. Communication between the edge layer and the client layer (i.e., MT) is implemented by using WebSocket technology [38].

In addition, the coarse-grained information about the area of interest (i.e., indoor floor plan) is of great importance to guarantee system performance. The system requires

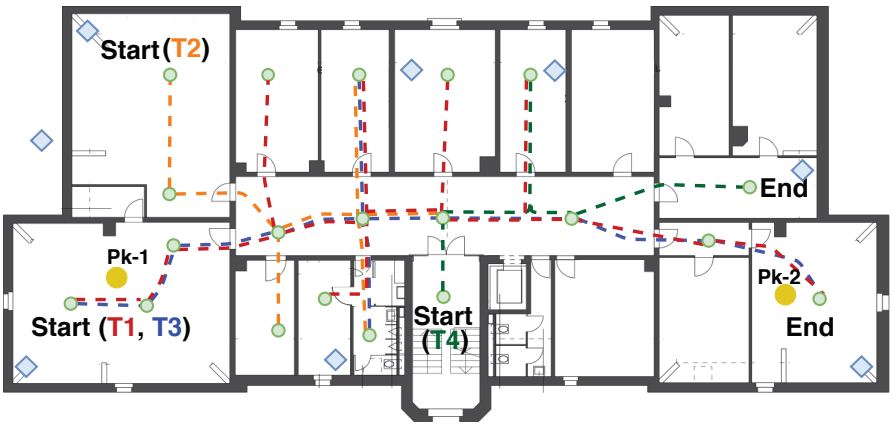


Figure 5.9: Trajectories definition in Scenario 1, circle yellow points are the kidnapped robot check points.

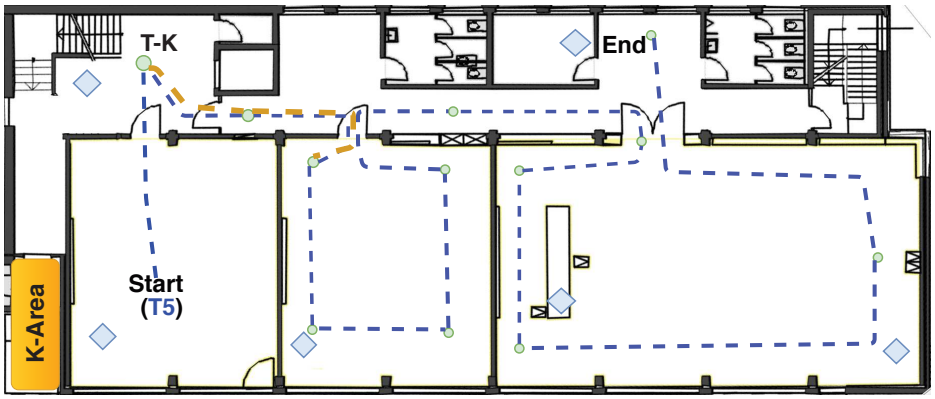


Figure 5.10: Trajectories definition in Scenario 2, yellow region is the area where particles are kidnapped.

information related to physical connections among zones, i.e., connectivity among zones. We define 14 zones for scenario 1 and 7 zones for scenario 2 in our areas of interest (details of the scenarios can be found in Figures 5.5 and 5.7). Each zone is a wall separated subarea (e.g., corridor, rooms). The transition model is defined based on the zone distribution. Figures 5.6 and 5.8 show the physical layout of the indoor environment, zone distribution and the transition model built on top of the indoor layout. The probability transition matrix T is set based on the assumption that the probability of staying in the same zone is higher than transferring to another one.

To ensure independence between the individual learning methods in HMM-d, we set up three conceptually different machine learning algorithms (KStar, Multilayer Perceptron (MLP) and J48). Weka for Android library [52] was used to implement the individual machine learning algorithms at the Client layer. To build the zone fingerprinting database, we collected 11200 fingerprint instances, approximately 800 in each zone. The structure of a fingerprint instance consists of Wi-Fi RSS and MF readings. We ask a person to walk randomly through each zone holding the phone in her hand. Zone fingerprint database entries were collected equally distributed over the whole area in each zone. The data collection rate is only constrained by the computational capabilities of the Wi-Fi sensor of the MT. Thus, in our experiments, every fingerprinting entry was collected at a rate of 3 entries/second. Since our approach does not need to predefine any survey point, the time needed to build the fingerprinting database is proportional to the number of collected instances multiplied by the instance collection rate. Machine learning algorithms have internal parameters that are optimized during the training process. Nevertheless, some algorithms have internal parameters that are not optimized during the training process. These parameters are named hyperparameters, which have a significant impact on the machine learning algorithms' performance. Similar to Chapter 4, we use a nested cross validation approach to choose the optimized hyperparameter values [76]. Thus, we configured the MLP method to single hidden layer with 14 neurons in scenario 1 and 7 neurons in scenario 2. Activation function was configured to the sigmoid function [110]. For the J48 method, we configure *gini* as function to measure the quality of a split [12]. For the KStar method, global percent ratio was configured to 30%. To reduce the negative impact of environmental changes and different hardware, we use differential Wi-Fi RSS instead of absolute raw values [108].

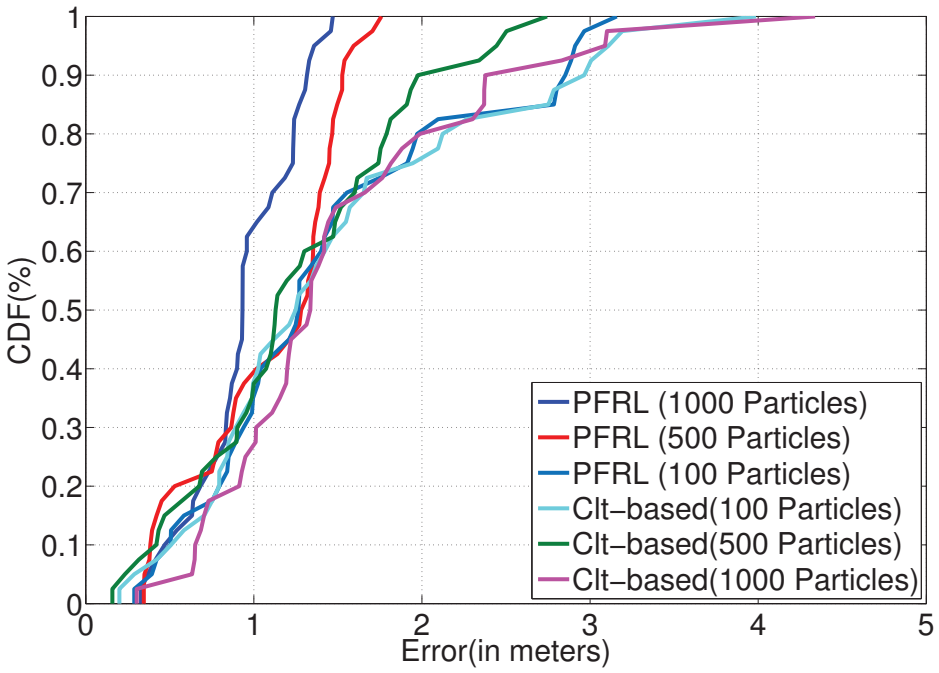


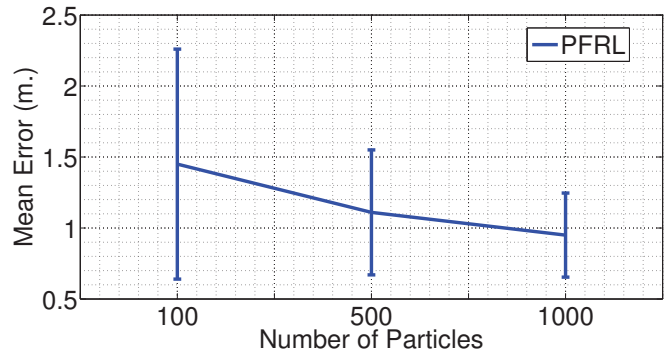
Figure 5.11: Impacts of particle numbers on performance of PFRL and client-based PFRL in scenario 1.

5.4 Performance Evaluation

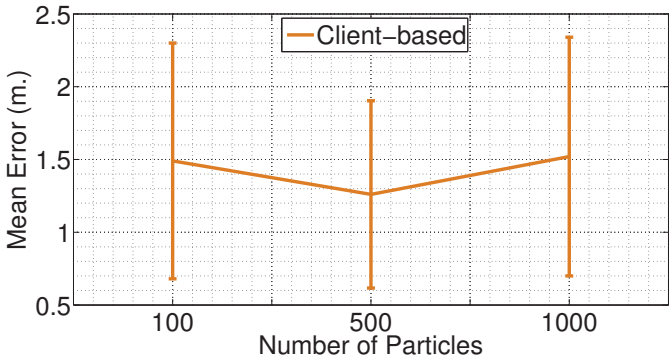
We made intensive experiments in two buildings of the Institute of Computer Science at the University of Bern. The first scenario is an office-like environment with an area of $702m^2$ ($39m$ x $18m$). The second scenario is a classroom-alike scenario with an area of $524m^2$ ($36.2m$ x $14.5m$). The MT is held by a person moving along five different trajectories. Every time when a new fingerprint measurement is available, the zone detection method is launched. Figures 5.9 and 5.10 show the physical layout of experiment areas, where trajectories are dotted lines, circle green points are the position checking points, diamond blue points are the anchor nodes.

5.4.1 Indoor Tracking Accuracy

In this section, we discuss the tracking performance of the PFRL system. Additionally, to show the benefits of our distributed localization approach using a two-layer architecture, we compare the performance between our two layer-based PFRL with a client-based version of PFRL, where all the computations are hosted on mobile



(a) PFRL (distributed) Confidence Interval.



(b) PFRL (client-based) Confidence Interval.

Figure 5.12: PFRL Confidence Intervals.

devices. Hereafter, we will refer to the two layer-based PFRL system as PFRL. The Client-based PFRL will be referred to as the client-based PFRL (Client-based). To evaluate the system performance, we consider the metrics of Cumulative Distribution Function (CDF) of localization errors, mean tracking error, the standard deviation of localization errors, and average processing time to get the position.

Localization Accuracy vs Number of Particles

Localization accuracy can be theoretically boosted by using more particles [50]. However, increasing the number of particles leads to increased computational complexity of the application too. A large number of particles produces computational inefficiency and high memory request. Figure 5.11 shows the CDF of localization errors for PFRL and client-based PFRL with different particle numbers in scenario 1.

PFRL achieves better performance when using 1000 particles compared to when using

100 and 500 particles. PFRL is able to reduce the mean localization error in a 24% by increasing the number of particles from 100 to 500, whereas by increasing the number of particles from 500 to 1000, PFRL reduces the mean localization error in 14%. As shown in Figure 5.12a, PFRL also reduces the confidence interval by increasing the number of particles. The standard deviation is reduced by 64% and 45% when increasing the number of particles from 100 to 500 and from 100 to 1000 respectively. Although the mean localization error of PFRL decreases by 34.5% when increasing from 100 to 1000 particles, the mean localization error improvement is only about 14% when particles are increased from 500 to 1000. Therefore, it is expected that after a certain number of particles, the mean localization error is not further improved.

Regarding the client-based approach, the best performance is achieved when 500 particles are used, which is better than using 100 and 1000 particles (also presented in Table 5.1). As shown in Figure 5.12b, client-based PFRL reduces the confidence interval by increasing the number of particles from 100 to 500. However, the confidence interval is increased when 1000 particles are used. To explain this behaviour, we look at the negative influence produced by increasing processing time in indoor real-time localization systems. The efficiency of real-time systems depends not only on the precise results but also on the latency of the system to compute the information. In real-time localization, high latency leads the system to stay processing a position while the ground truth position of the mobile device is constantly changing. Therefore, clearly in real-time localization applications, processing time influences the accuracy performance of the system. In the client-based method, we noticed that the average processing time seems to grow exponentially with the number of particles (see Figure 5.13). Therefore, there is no more performance improvement when a certain particle number is used, due to the negative influence of the exponential growth of the processing time.

Figure 5.13 shows the average processing time of the particle filter method for the client-based PFRL with different particle numbers. As we can see, when using 1000 particles, the average processing time is 290 ms, which is much bigger than the average processing time of 170 ms when using 500 particles. Therefore, due to the limited computation resources available on mobile devices, increasing the number of particles exponentially increase processing time, which leads to lower localization performance. This explains why 500 particles lead to better accuracy than 1000 particles when using a client-based solution.

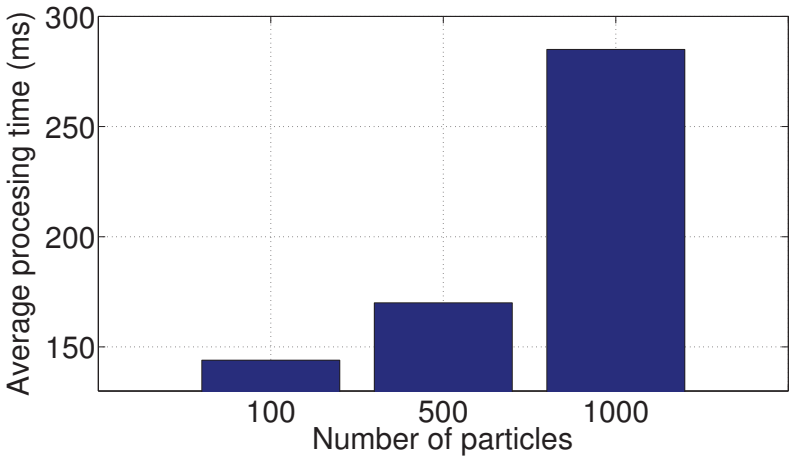


Figure 5.13: Processing time of client-based solution vs number of particles

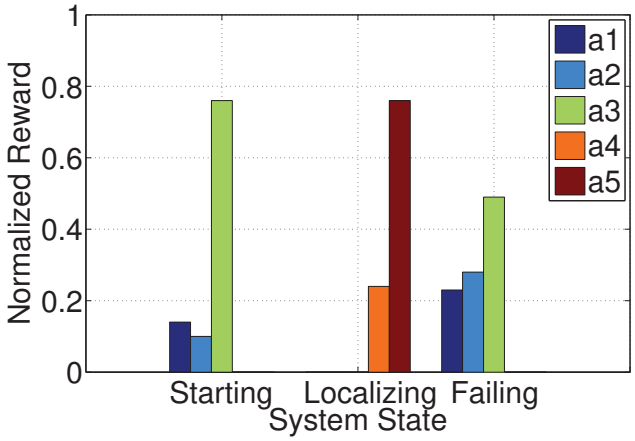


Figure 5.14: System State vs Normalized Rewards (Normalized Q-table)

Failure Avoidance and Recovery Performance

The global localization and the kidnapped robot problem are used to evaluate the ability of the system to avoid, detect, and recover itself from localization failures. Moreover, this evaluation measures the ability of self-localizing the target when the system is started (i.e., global localization problem). To test the recovery performance of PFRL, we conducted two experiments in this section. The first experiment tests the self-localizing ability of the system when it is started. We refer to this experiment as the Global localization experiment. The second experiment tests the system ability to recover itself from failures when the system is in normal operation (Localizing). We refer to this as the Kidnapping robot experiment.

The **Global localization experiment** was performed in scenario 1. In this experiment, we simulate localization failures by setting up the initial position of the set of particles at an arbitrary position outside the area of interest. Afterwards, the pedestrian started the system standing at a known position (these positions are shown in Figures 5.9 and 5.10). We repeated the experiment in three different zones. The derived position was registered in each iteration. Figure 5.14 shows the normalized rewards (normalized rewards from Q-table) learned in each state. It can be seen that the reinforcement learning method defines *e3* as the best action to perform at states Starting and Failing, and action *e5* is defined as the best action to be executed when the system is at Localizing state. Therefore, to recover the system from localization failures, particles are spread based on the zone probability distribution predicted by the HMM-d method. To avoid localization failures, the 10% worst evaluated particles are spread based on the zone probability distribution predicted by the HMM-d method.

As it can be seen in Figure 5.15, the average number of iterations to recover the system from localization failures is 4. Each iteration is processed when new Wi-Fi information is available. Since our MT has a Wi-Fi sampling rate up to 3Hz, the latency to recover the system is approximately $1.5s$. This means that the initial position is determined at approximately $1.5s$ after the system is started. Moreover, if a localization failure occurs during the tracking process, the system can be automatically recovered with an acceptable time latency of $1.5s$. Regarding the number of iterations, this current approach outperforms by 70% to the equally distributed (ED) localization recovery method presented in [17].

To present PFRL's capability to generate fast particle convergence, we show the physical locations of particles during a localization procedure. First, we present the zone

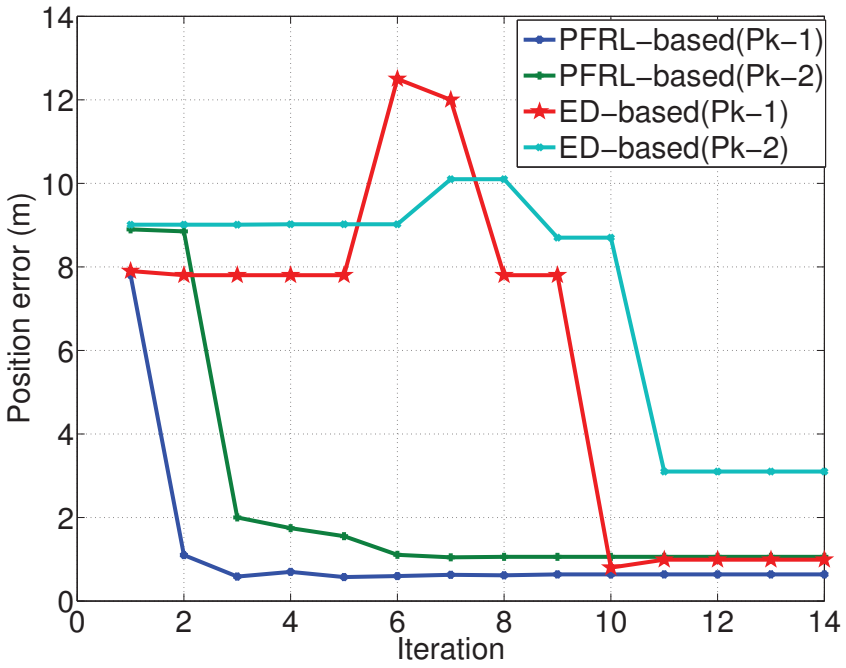


Figure 5.15: Scenario 1: Global Localization Failure recovery. Particles convergence time after localization failure

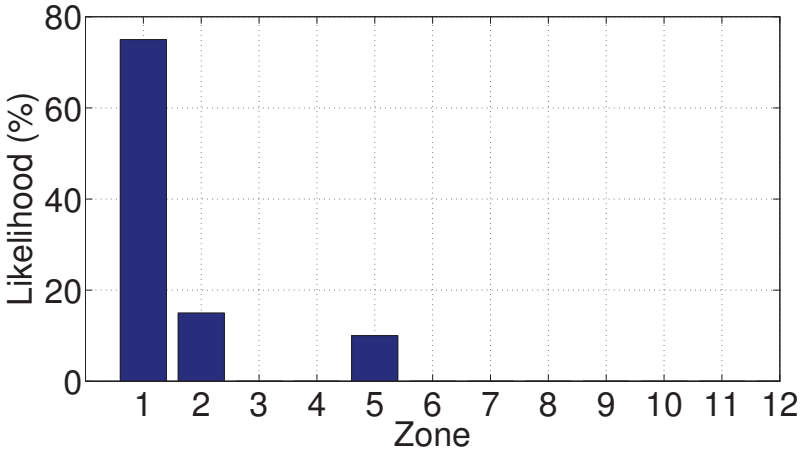


Figure 5.16: HMM-d Zone probability distribution during the Global localization experiment in scenario 1.

probability distribution results of HMM-d, which has a consequence on the particle convergence speed. Figure 5.16 shows the zone distribution when the system was started at Pk-1 position during the Global localization experiment. According to the normalized rewards table (Figure 5.14) and the zone probability distribution (Figure 5.16), 75% of the particles are distributed over zone 1, whereas 15% and 10% of the particles are distributed over zone 2 and 5 respectively. Thus, more particles are generated in the zones with high likelihood of containing the ground truth location. Unlike ED-based, PFRL focuses on the exploration of high-probability zones. Figures 5.17, 5.18, and 5.19 show the physical distributions of the particles in the Starting state after a localization failure recovery procedure is triggered. The ground truth position is located at Pk-1 (see Figure 5.9). The cyan points represent the particles of the PFRL method, whereas the red points depict particles of the ED-based method. As mentioned, in PFRL particles are distributed based on the normalized rewards table (Figure 5.14). In the Starting state, the particles are distributed based on action $e3$. Therefore, to recover the system from the global localization problem, particles are distributed according to the zone probability distribution given by the zone prediction method HMM-d. Consequently, particles in PFRL converge faster than ED-based method, which leads to faster failure recovery in PFRL.

Figures 5.17, 5.18, and 5.19 show the particle distributions in PFRL and ED-based approaches. The cyan points represent PFRL particles; the red points represent ED-based particles; the diamond yellow points represent the ground truth position. In

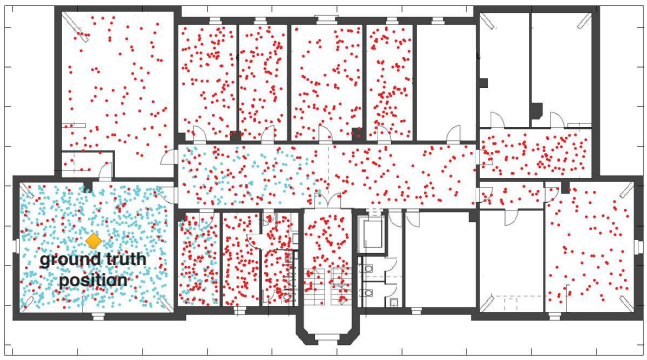


Figure 5.17: Particles distributions after 0.5 s when a localization failure recovery happens

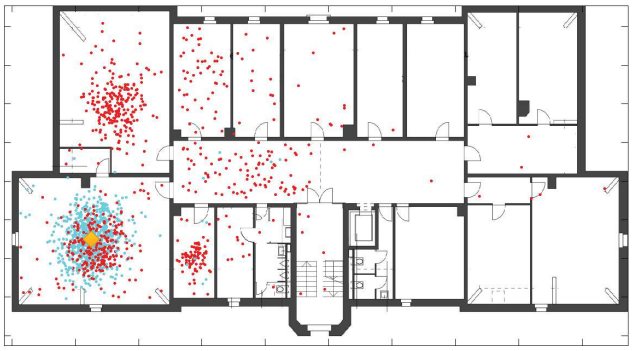


Figure 5.18: Particle distributions after 1.2 s when a localization failure recovery happens

Figure 5.19: Particles distributions after 1.6 s when a localization failure recovery happens

the PFRL method, the particles converge faster than the ED-based method.

The **Kidnapping robot experiment** was performed in scenario 2. In this experiment, we simulate localization failures by kidnapping all the particles to a predefined area in the environment. We refer to this area as the K-area (yellow region in the left bottom corner in Figure 5.10). After 25 seconds in normal localization operation along trajectory T-K (trajectory T-K and K-area are depicted in yellow color in Figure 5.10), the system is kidnapped to the K-area. We registered the mean localization errors and time to recover the system from the localization failure. We compare the recovery performance of PFRL to ED-based [17] and PF-based [20]. Figure 5.20 depicts mean localization error over time for the three tested localization methods in scenario 2. PFRL recovers around 5 seconds after the failure. ED-based approach recovers around 11 seconds after the failure. PF-based (without recovery) solution never recover from the failure. It is worth to mention that the main difference between the Global localization and Kidnapping robot experiments is that in the latter the system must detect the failure to execute the recovery method. In the Global localization experiment, the system is aware that it is starting. Thus, the recovery method is launched immediately after the system starts. Therefore, in the Kidnapping robot experiment, we test the performance of the System State Detection Method presented in section 5.2.5 and the Reinforcement Learning method for robust tracking resampling presented in section 5.2.5. As it can be seen in Figure 5.20, PFRL overcomes by around 50% to the ED-based method. Unlike ED-based, PFRL implements an effective method to detect localization failures. This method is based on the HMM-d method for zone prediction. Thus, if any particle is placed in the current predicted zone, the system assumes a failure. Thus, a kidnapping robot problem is detected immediately when it occurs. Moreover, PFRL implements an effective reinforcement learning-based method for recovering the system from failures. Therefore, after detecting a failure, the system is recovered by sampling particles based on the normalized rewards (see Figure 5.14) that are learned by the reinforcement learning method. This allows to detect and recover the system in around 5 seconds.

Localization Performance comparison with other systems

It is difficult to fairly compare our approach with other state-of-the-art localization approaches (e.g., fingerprinting-based, landmark-based, range-based). This is because indoor localization system performance is environmental-dependent (i.e., they rely on the presence of numerous landmarks in the environment), and it is impossible

Table 5.1: Scenario 1: Localization methods performance

Configuration	Mean error	Standard Deviation	90% Accuracy
PFRL (100 Ptc.)	1.45m	0.81m	2.9m
PFRL (500 Ptc.)	1.11m	0.45m	1.5m
PFRL (1000 Ptc.)	0.97m	0.3m	1.3m
Client-based (100 Ptc.)	1.493m	0.907m	3.1m
Client-based (500 Ptc.)	1.267m	0.645m	2.0m
Client-based (1000 Ptc.)	1.515m	0.8188m	2.9m
PF-based (1000 Ptc.)	1.15m	0.61m	2.1m
NLST	3.79m	2.52m	8.0m
k-NN (k=3)	3.32m	1.89m	6.1m
Kalman Filter	3.36m	1.11m	4.1m

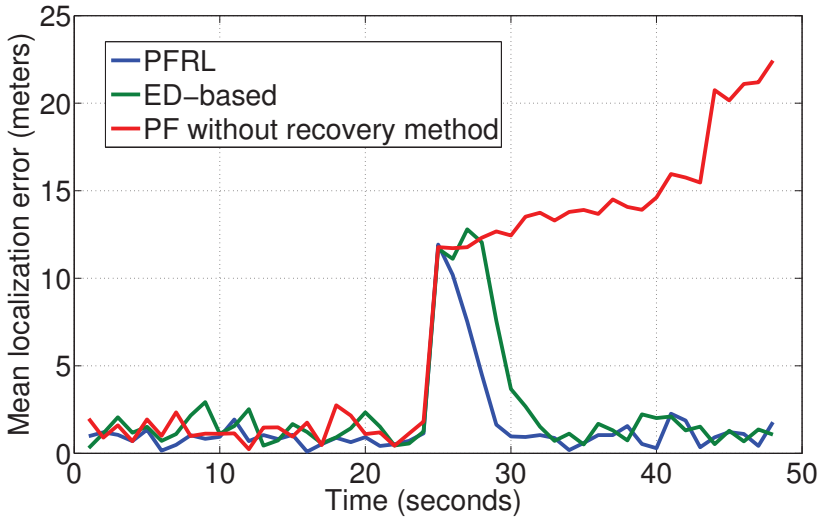


Figure 5.20: Scenario 2: Kinapping Robot Problem. Particles convergence time after a localization failure is detected.

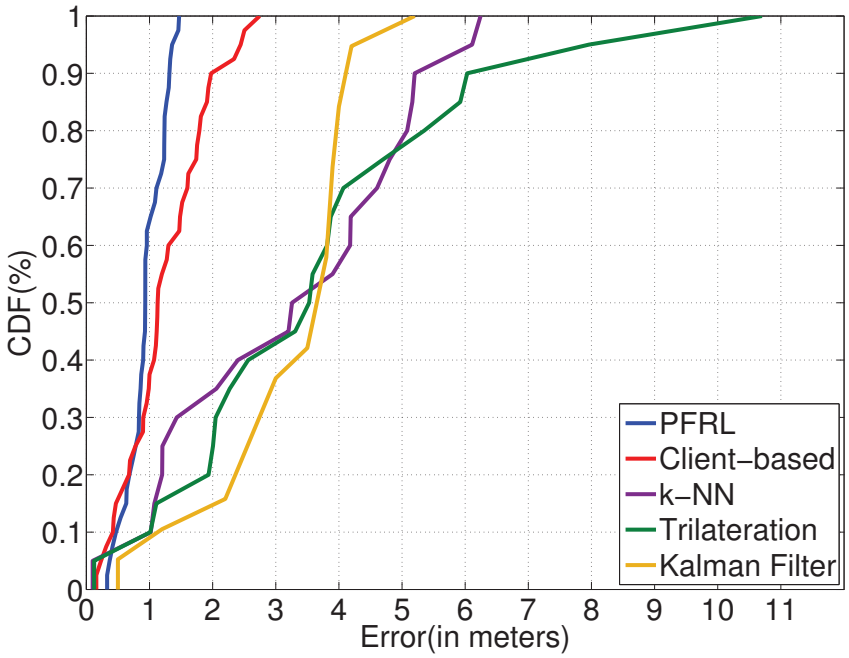


Figure 5.21: Scenario 1: Localization error CDF of PFRL (1000 particles), Client-based (500 particles) NLS, k-NN (k=3) and KF.

to duplicate the exact indoor environments in another indoor areas. Moreover, it is rather hard to implement all the specific details of an existing solution and repeat the identical experiment to get the same results that were collected in another physical indoor environment. Similar to other localization systems, we compare the performance of our localization approach with the k-Nearest Neighbors (k-NN), Nonlinear Least Squares Trilateration (NLST), and Kalman Filter-based (KF) localization methods. Morevover, we compare PFRL to another basic particle filter-based (PF-based) approach which is presented in [20]. The KF localization method was implemented as part of a bachelor thesis project [6].

Figure 5.21 shows the CDF of localization errors for the best performance of PFRL, Client-based, PF-based, k-NN (k=3), NLST, and KF methods. Table 5.1 summarizes the results, which show that NLST achieves the worst localization performance with around 8.0m for 90% accuracy. PFRL overcomes NLST by approximately 83.7% and 74.4% regarding 90% accuracy and mean error respectively. Moreover, the standard deviation of PFRL is 88.09% smaller than NLST. k-NN achieves around 6.1m for 90% accuracy with the mean error of 3.32m and the standard deviation of 1.89m. The KF approach achieves a 90% accuracy of 4.1m, the mean error is 3.36m and the standard

Table 5.2: Scenario 2: Localization methods performance

Configuration	Mean error	Standard Deviation	90% Accuracy
PFRL (1000 Ptc.)	0.98m	0.49m	1.8m
Client-based (1000 Ptc.)	1.3m	0.65m	2.1m
PF-based (1000 Ptc.)	1.34m	0.7m	2.1m
NLST	4.57m	1.9m	7.0m
k-NN (k=3)	3.83m	1.71m	6.1m
Kalman Filter	3.29m	1.24m	4.9m

deviation is $1.11m$. Therefore, PFRL overcomes k-NN by around 78.68%, and KF by around 68.29% considering 90% accuracy. The mean error of the PFRL approach is 71.1% and 70.78% better than for KF, and k-NN respectively. Experiment results show that PFRL outperforms Client-based, KF, NLST, and k-NN for accuracy and stability. Although PF-based achieves high localization performance, PFRL outperforms PF-based by around 15.7% and 24.6% considering mean localization error and standard deviation respectively. This is because PF-based does not have any method to identify if the AN and the mobile client are located at the same zone. Thus, the same ranging method is adopted for all the ANs in PF-based method. Unlike PF-based, PFRL includes a zone prediction method, which supports to choose the proper ranging model for each zone. Therefore, PFRL outperforms traditional particle filter and fingerprinting-based localization methods (e.g., k-NN) by combining range-based localization methods and fingerprinting models.

Performance vs Area of Interest

To validate the environment independence of PFRL, we chose a second scenario to deploy the localization system. As mentioned in previous sections (Section 5.4), scenario 2 is a classroom-alike indoor scenario at the University of Bern with an area size of $524m^2$. We set up PFRL with the configuration that achieved the best performance on experiments executed in scenario 1. Figure 5.10 depicts scenario 2 and trajectory 5, which was used to test the localization approaches. Table 5.2 summarizes the mean tracking error, standard deviation and 90% accuracy. Figure 5.22 shows the CDF of localization errors for the best performance of PFRL, Client-based, PF-based, k-NN (k=3), NLST and KF methods. PFRL achieves around $0.98m$ for mean localization error, which outperforms Client-based, PF-based, NLST, K-NN and KF by around 24.6%, 24.8%, 78.5%, 74.1% and 70.2% respectively. Although the high localization performance of PFRL observed in scenario 2, the mean error, standard

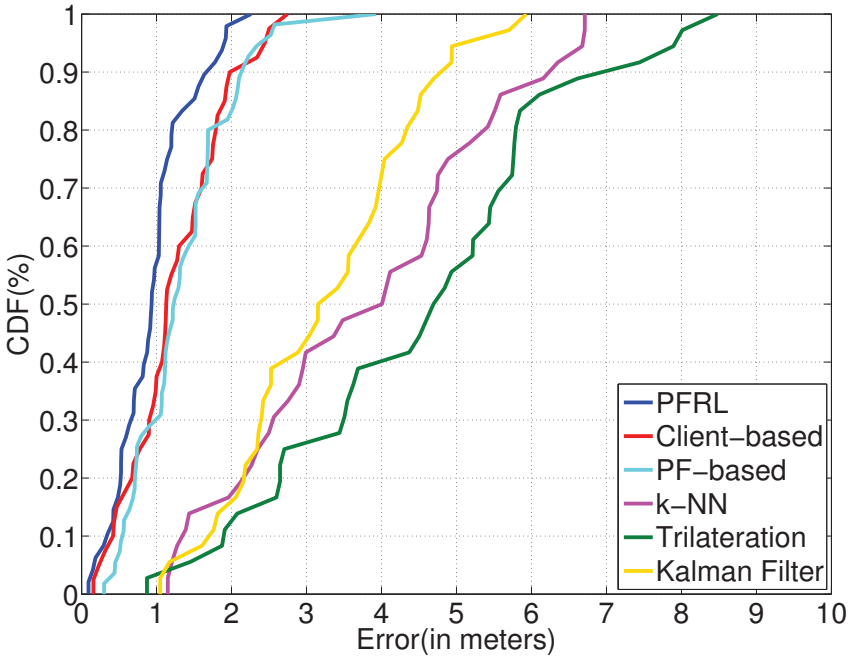


Figure 5.22: Scenario2: Localization error CDF of PFRL (1000 particles), Client-based (500 particles) NLS, k-NN and KF.

deviation and 90% accuracy were slightly increased compared to experiments in scenario 1. This reflects that the density of ANs along the area of interest influence the localization performance. In scenario 1 we deployed 8 ANs, whereas in scenario 2 we deployed 7 ANs.

5.4.2 System Complexity Analysis

In this subsection, we discuss the theoretical time complexity of our system, which mainly comes from the ensemble learning models for zone prediction, particle filter approach for data fusion, and the reinforcement learning method for particle distribution. Thus, the theoretical time complexity of the system is the prediction time complexity of the zone prediction method plus the time complexity of the particle filter method plus the execution time complexity of the reinforcement learning method. The prediction time complexity of the ensemble learning model and time complexity of the particle filter method for indoor tracking are discussed in Chapter 4, and Chapter 6 respectively. Therefore, in the following paragraphs, we focus on the execution time complexity of the reinforcement learning method for particle distribution.

As mentioned in subsection 5.2.5, Q-learning algorithm is adopted to implement the reinforcement learning method. In Q-learning, the learning process is performed based on the Bellman equation (Equation 5.7). Thus, each iteration of Q-learning consists of an action selection process and a $Q(s,e)$ value update process. Q-learning algorithm cannot remember information other than what is stored in the $Q(s,e)$ variable. Access to $Q(s,e)$ information is restricted to information of the current state of the agent. Thus, time complexity of the action selection process is $O(N_e)$, where N_e is the number of element in the set of actions. After the execution of an action, Q-learning algorithm can only propagate information from the new state of the agent to the previous state of the agent. Thus, time complexity of the value update process is $O(N_s)$, where N_s is the number of states. Therefore, the execution time complexity of the reinforcement learning method for particle propagation can be expressed as $O(N_s + N_e)$. Therefore, the execution time complexity of the reinforcement learning method grows linearly regarding the number of states and number of actions.

Since the prediction time complexity of the ensemble learning method for zone prediction and time complexity of the particle filter grow exponentially, the execution time complexity of the reinforcement learning method does not increase the overall time complexity of the localization system. Therefore, time complexity of the PFRL localization approach grows exponentially regarding the number of extracted features, number of training samples, number of particles, and number of ANs used in the localization process.

5.5 Conclusions

In this chapter, we proposed a distributed architecture for indoor tracking. We presented a MEC architecture approach, where lightweight methods run on the mobile target device and heavy computations are offloaded to an edge server. Thus, by exploiting MEC architecture, we proposed a particle filter-based reinforcement learning (PFRL) approach for autonomous robust wireless indoor tracking. Our approach was validated on a distributed machine learning-based network architecture, which includes a client layer and an edge layer. The client layer includes mobile devices that host lightweight ML algorithms (supervised ML algorithms) to recognize zones, while the edge layer includes edge server that hosts heavy machine learning operations to run complex particle filter and reinforcement learning calculations. The PFRL algorithm includes several components. An efficient ensemble predictor that

could achieve high zone prediction performance by integrating HMM with discriminative learning techniques. A reinforcement learning approach is applied on top of the proposed particle filter to improve the system robustness against positioning failures. We evaluated our localization system in two complex real-world indoor environments. Evaluation results show that our proposed method can deliver more accurate localization results and is more robust to localization failures than traditional indoor localization methods. Thanks to the reinforcement learning approach, the proposed PFRL solution could make the localization system converge much faster than other systems without a failure recovery mechanism.

Since time complexity of the proposed reinforcement learning method grows linearly, the execution time complexity of the reinforcement learning method does not increase the overall time complexity of the localization system. Moreover, we compared the PFRL localization performance to a client-based PFRL approach. Results have shown that the localization performance of PFRL overcomes the localization performance of the client-based approach. This demonstrated the negative influence produced by limited processing resources of mobile devices when increasing the processing time complexity of the localization algorithms.

6

MEC based Indoor Tracking for Wireless Internet of Things Mobile Devices

6.1 Introduction

The current expansion of the IoT domain has attracted significant attentions to context-aware applications for IoT mobile devices. IoT involves extending internet connectivity beyond standard devices, such as laptops, smartphones, and tablets, to any range of computing devices that connect to a network and have the ability to transmit data. These devices must be able to communicate and interact over the Internet, and they can be remotely monitored and controlled.

Typically, indoor localization applications in smartphones aim to report the current position directly to the smartphone user. It is done locally by using the user interface of the smartphone (i.e., screen). However, IoT devices intend to report their current state (e.g., position, temperature, etc.) to a management monitoring application. Thus, IoT applications must be based on a suitable infrastructure to allow remote access monitoring to IoT devices information. In this chapter, we present a three

layer MEC-based indoor tracking system (InTrack) for IoT mobile devices. Figure 6.1 shows the architecture of the system. By exploiting MEC architecture, we extend the localization approach presented in Chapter 5 to provide localization and tracking services to mobile devices in the IoT domain.

Our proposed localization approach includes a Cloud layer, which is responsible for remote monitoring and the storage of historical localization information of IoT devices. Moreover, we include a PDR method to estimate the velocity of mobile devices and a ranging method to provide time of flight based ranging estimation.

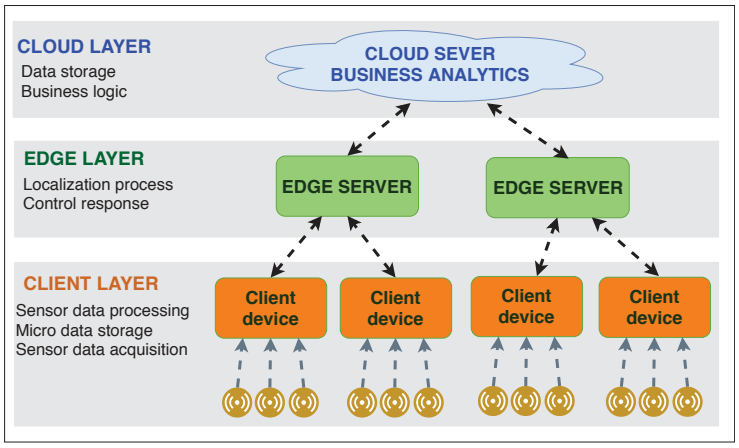


Figure 6.1: MEC-based Indoor Tracking System Architecture.

The rest of the chapter is organized as follows. The architecture of our proposed tracking system is reviewed in Section 6.2. Section 6.3 presents the implementation of the MEC-based tracking system. Section 6.4 discusses the performance evaluation results. Section 6.5 concludes the chapter.

6.2 System Architecture

This section presents the design details of the proposed MEC-based indoor localization system for IoT devices. Figure 6.2 summarizes the system layer architecture, which includes three layers: Cloud layer, Edge layer, and Client layer. Details of each layer are described in the following subsections.

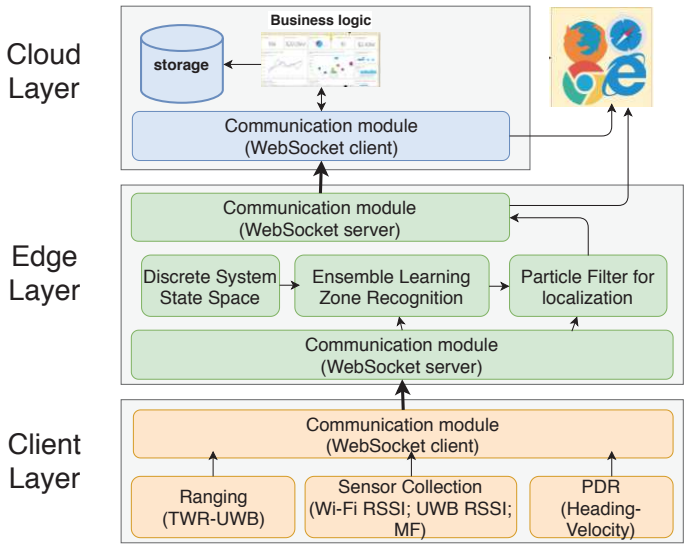


Figure 6.2: Layer Architecture of MEC-based Indoor Tracking System.

6.2.1 Client Layer

The client layer includes IoT mobile devices (MD) that are to be located. The mobile devices constantly collect data from on-board sensors, such as inertial measurement units, Wi-Fi or UWB radio interfaces, etc. Instead of sending raw data to the Edge layer directly, MDs process them locally to derive meaningful insights (e.g., movement directions and speeds, Wi-Fi and UWB fingerprints, and ranges) through three modules of PDR, fingerprint acquisition, and UWB-ranging. The derived information is then sent to the Edge layer via the data transmission module for further processing. This architecture leaves all the device-dependent data processing, such as Wi-Fi or UWB signal processing, to happen at the Client layer. A significant advantage of this design is that the Edge layer is completely independent of the client device specifications, which makes the system capable to support different IoT device types. For instance, a smartphone or Rasperry Pi can be easily integrated into the system, without any modifications at Edge and Cloud layers, as far as relevant information can be generated from raw data. Figure 6.4 shows the client layer architecture, whose core subcomponents are described in the following subsections.

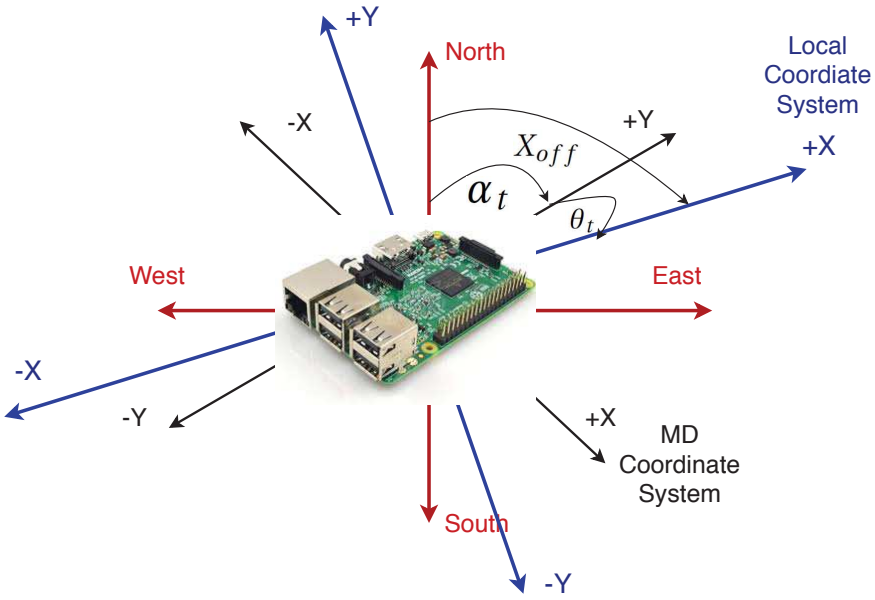


Figure 6.3: Angle Relation

PDR-velocity

In order to estimate the velocity of mobile devices, we use the accelerometer gyroscope, and the magnetometer sensors, from which the heading direction and the speed can be computed. To estimate the heading direction, we rely on a digital compass developed from the magnetometer, gyroscope, and accelerometer sensors embedded in the MD. The heading direction estimation process is introduced in chapter 3.

To estimate the speed of the MD, we use the accelerometer sensor. Thus, the speed is computed by using Equation 6.1

$$\overline{v} = \int_{t_o}^{t_f} a \cdot dt, \tag{6.1}$$

Since accelerometer data contains huge non-zero mean noise, accelerometer data is smoothed by using low pass filters. The PDR-velocity method was implemented as part of a bachelor thesis project [111].

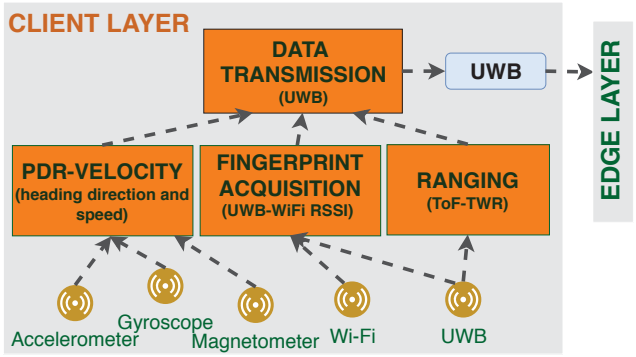


Figure 6.4: Client layer architecture

Fingerprint acquisition

Through the embedded radio interfaces, the mobile device collects radio frequency fingerprint values from the surrounding environment. Afterwards, this data is passed to the Edge layer as inputs for the zone recognition process.

Two Way Ranging

The ranging process is conducted by the Two Way Ranging method (TWR). TWR is a method where radio transmissions are timed between two devices and combined mathematically to estimate the distance between the devices. Thus, TWR determines the time of flight of the radio frequency signal and then computes the distance between the nodes by multiplying the time by the speed of light [95]. Details of the TWR method can be found in section 2.4.1

Data transmission

Latency minimization is an underlying requirement of context-aware services. Therefore, the communication method must allow low data transfer latency from and to the server. WebSocket is a communication protocol, providing full-duplex real-time communication channels over a single TCP connection [38]. WebSockets provide standardized methods to transfer data from the server to clients without being first requested by the client. This allows data to be sent back and forth while keeping an open full-duplex connection between server an client. Thus, when PDR-velocity, sensor collection, and ranging are completed, the output data from these processes are transmitted to the Edge layer for further processing.

6.2.2 Edge Layer

The Edge layer is responsible for running the computation-heavy localization algorithm to calculate indoor locations. It includes three sub-modules and two interfaces. The zone prediction module is to estimate the indoor zone information using the received fingerprints. The space representation module constrains the location estimation ranges. The data fusion module applies advanced particle filter to fuse multiple inputs to estimate locations. The required information to feed zone prediction and data fusion modules are periodically received from the Client. Calculated indoor locations are sent to the Cloud layer via a web-socket. Figure 6.5 summarizes the processes of the Edge layer. Details of each component are given below.

Zone Prediction

Zone prediction results are provided by the HMM-d method. As mentioned in Chapter 4, the key idea of HMM-d is to combine conceptually different individual machine learning models in an HMM. Thus, we combine some individual machine learning algorithms to improve prediction performance compared to individual models. Design and implementation details about the HMM-d method can be found in Chapter 4.

Space Representation

To minimize the algorithmic complexity, our system defines a discrete structure to replace the conventional floor map. All the system states (i.e., indoor positions) are represented by a discrete set of locations by converting from a continuous state space to a discrete state space. Therefore, we consider the physical environment as a spatial data structure that defines space as an array of cells arranged in rows and columns. Thus, each cell (i.e., location) belongs to a zone. We define two types of zones, enabled zones and not enabled zones. In the enabled zones the target object is allowed to move (e.g., rooms, corridors). In the not enabled zones the target object is not allowed to move (e.g., through walls). Therefore, the Space Representation component computes the map likelihood (i.e., allowed areas to spread particles), and the transition model (i.e., connections among zones). The discrete space representation model is introduced in Chapter 3.

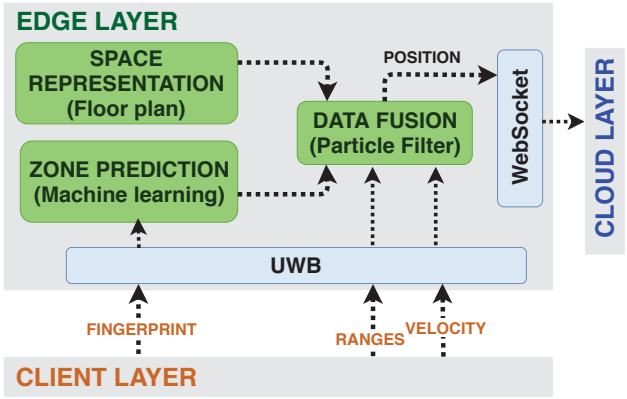


Figure 6.5: Edge layer architecture

Data Fusion

We consider indoor localization as a filtering problem, in which the position of the target can be computed from several noisy environmental observations. Thus, this work focuses on a particle filter approach to provide indoor localization.

In a particle filter approach, a belief of the target position is computed based on the observations, (i.e., posterior probability distribution). The posterior probability can be represented as a set of weighted particles. Particle filters estimate the posterior probability distribution of the system state based on some measurements q_t at time t [20]. At time t, the system state vector X_t is defined by Equation 2.39.

Since locations belong to a zone, zone z_t can be computed from the current Cartesian coordinates (x_t, y_t). Therefore, function $z_t = f(x_t, y_t)$ derives the current zone z_t. Thus, z_t can be written as:

$$z_t = getZone(x_{t-1} + \ell_t \cdot cos(\theta_t), y_{t-1} + \ell_t \cdot sin(\theta_t)) \tag{6.2}$$

Therefore, the particle filter prediction function can be written as:

$$X_t = \begin{pmatrix} x_{t-1} + \ell_t \cdot cos(\theta_t) \\ y_{t-1} + \ell_t \cdot sin(\theta_t) \\ \theta_t \\ \ell_t \\ z_t = getZone(x_t, y_t) \end{pmatrix} \tag{6.3}$$

Both θ_t and ℓ_t values are calculated by PDR methods, whereas z_t is a discrete random variable that identifies the zone where the particle is located at time t. State vector X_t^i of each particle is updated from the particles at the previous time interval X_{t-1}^i based on Equation (6.3). Thus, the new set P_t is calculated from P_{t-1}. Particles are allowed to move only through non-restricted areas, e.g., movement through walls is not allowed.

After updating particles using Equation (6.3), the associated weight w_t^i of the propagated particles must be corrected. The associated weight update is based on the likelihood of the observations conditioned on each particle state $P(q_t \mid X_t^i)$ at time t. The observation vector is defined by the estimated ranges to different ANs and the estimated zone information. Thus, the observation vector at time t is defined as $q_t = [\hat{d}_t, \hat{s}_t]$, where $\hat{d}_t$ contains ranges to different ANs and $\hat{s}_t$ contains the observations related to the predicted zone.

Since the ranging method (i.e., the method to estimate ranges) and the zone prediction method are different, we assume that range and zone prediction information are independent of each other. Therefore, the probability $P(q_t \mid X_t^i)$ can be determined as follows:

$$P(q_t \mid X_t^i) = P(d_t \mid X_t^i) \cdot P(s_t \mid X_t^i) \tag{6.4}$$

Hereafter, we refer to $P(d_t \mid X_t^i)$ as the ranging likelihood, and $P(s_t \mid X_t^i)$ as the zone likelihood. The associated weight w_t^i of each particle is given by the ranging and zone prediction information. The particle at the absolute position (x_t, y_t) with low probability to observe d_t^j will be assigned a small ranging likelihood. Particles positioned at zones with low probability of observing s_t will be assigned small zone likelihood values.

Since ANs are programmed to operate independently, we can assume that the ranges to different ANs are independent from each other. Therefore, the ranging likelihood can be calculated by using Equation 3.5.

Zone likelihood refers to the probability of observing s_t in the current particle state X_t^i. Therefore, $P(s_t \mid X_t^i)$ can be written as:

$$P(\hat{s}_t \mid X_t^i) = \frac{P(X_t^i \mid \hat{s}_t) \cdot P(\hat{s}_t)}{P(X_t^i)}, \tag{6.5}$$

where $\hat{s}_t$ is the zone related set of observations at time t. Since the $P(X_t^i)$ and $P(\hat{s}_t)$ are constant, $P(s_t \mid X_t^i)$ depends only on $P(X_t^i \mid \hat{s}_t)$. Therefore, $P(s_t \mid X_t^i) \propto P(X_t^i \mid \hat{s}_t)$. Applying Equation 6.2, $P(X_t^i \mid \hat{s}_t)$ can be written as follows:

$$P(z_t^i \mid \hat{s}_t) = \frac{P(\hat{s}_t \mid z_t^i) \cdot P(z_t^i)}{P(z_t^i)} \tag{6.6}$$

Since O is the set of observations related to the zone prediction process (see Equation 4.5), we can define $\hat{s}_t$ as an element of O ($\hat{s}_t \in O$). Therefore, $P(z_t^i \mid \hat{s}_t)$ is computed by the zone prediction method.

Algorithm 5 PF-MLM

Result: Mobile client position

Calculate the initial zone probability distribution: $p(z_{l_0} \mid q_{l_0}) = HMM\text{-}d()$

Distribute particles based on $p(z_{l_0} \mid q_{l_0})$

Initialize particle's weights: $W_0^i = 1/N_p$, $i = 1, 2, ..., N_p$ **while** *True* **do**

Update the particles: $X_t^i = G \cdot X_{t-1}^i + \eta$

Calculate the ranging likelihood: $P(\hat{d}_{j,t} \mid X_t^i) = \frac{1}{\sigma_j \sqrt{2\pi}} \exp^{-\frac{[\hat{d}_{j,t} - \sqrt{(x^i - x_j)^2 + (y^i - y_j)^2}]^2}{2\sigma_j^2}}$

Calculate the zone probability distribution: $p(z_{l_t} \mid q_{l_t}) = E\text{-}HCP()$

Calculate the zone likelihood: $P(z_{l_t} \mid X_t^i) = \frac{P(X_t^i \mid \hat{z}_{l_t}) \cdot P(\hat{z}_{l_t})}{P(X_t^i)}$

Compute unnormalized weights: $\hat{w}_t^i = P(X_t^i \mid \hat{z}_{l_t}) \cdot \prod_{j=1}^{M} P(\hat{d}_{j,t} \mid X_t^i)$

Normalize weights: $w_t^i = \frac{\hat{w}_t^i}{\sum_{n=1}^{N} \hat{w}_n^i}$

Resample the particles

Compute the estimated position: $X_t = \sum_{i=1}^{N} w_t^i \cdot x_t^i$

end

6.2.3 Cloud Layer

The Cloud layer is responsible for remote monitoring of mobile devices and the storage of historical localization information. This information is related to users and the localization process along with multiple areas of interest. The information is stored in

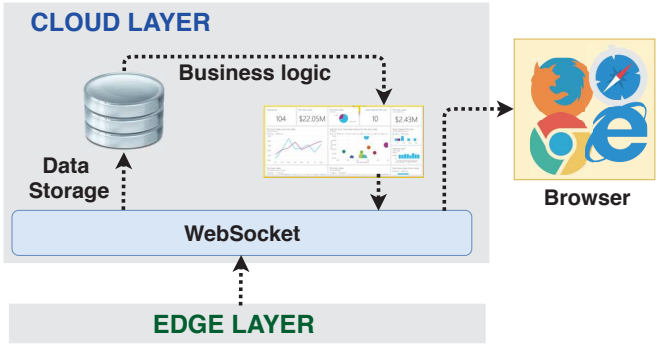


Figure 6.6: Cloud layer architecture

a structured database in the Cloud server. Thus, the Cloud layer enables high-order queries over the historical localization information to provide predictive analysis and business control. Therefore, allowing data collection from multiple scenarios and mobile devices (i.e. clients) and making this data accessible anywhere in the world are the main advantages of the Cloud layer.

Client devices gather data from on-site, then they pass this data to the Edge layer for processing (i.e., localization). Processed data is then passed to the Cloud layer, which is typically in a different geographical location. Thus, the cloud layer benefits from client devices by receiving their data through the other layers. Figure 6.6 shows the internal architecture of the Cloud layer.

6.3 Implementation

Our MEC-based tracking system comprises five main components: a client mobile device (MD), some commercial Wi-Fi access points (Wi-Fi-AN), some UWB anchor nodes (UWB-AN), an Edge server (ES), and a cloud server (CS). The MD is the device to be localized. Positions of UWB-AN are chosen to provide the maximum coverage inside the area of interest. Table 6.1 summarizes the specification of each component. Communication between the Cloud and the Edge layer was implemented by using WebSocket technology. WebSocket is a computer communication protocol, which allows two or more connected devices to communicate with one another in both directions through a single TCP connection. It is supported by many platforms. WebSocket technology uses the HTTP upgrade header to change from the HTTP to the WebSocket protocol [38]. Thus, Tornado [103] was used to provide web server and WebSocket server in the cloud layer.

Table 6.1: Fog-edge localization components

Layer	Component	Specifications
Cloud	**Cloud Server**	**Model:** HP EliteBook **CPU:** 2.30 GHz Intel Core i5-5300U **OS:** Windows 10 Enterprise **RAM:** 8 GB
Edge	**Edge Server**	**Model:** HP EliteBook **CPU:** 2.30 GHz Intel Core i5-5300U **OS:** Windows 10 Enterprise **RAM:** 8 GB **UWB Interface:** Sequitur Pi (InGPS litle)
Client	**Client Device**	**Model:** Raspberry Pi Model B **CPU:** Quad Core 1.2GHz **OS:** Raspbian 4.14 **WLAN:**WiFi b/g/n **UWB-IMU:** Sequitur Pi (InGPS litle)
	Wi-Fi-AN	**Model:** D-Link (D-635 and DAP-2553)
	UWB-AN	**Model:** Raspberry Pi Model B **CPU:** Quad Core 1.2GHz **OS:** Raspbian 4.14 **WLAN:**WiFi b/g/n **UWB:** Sequitur Pi (InGPS litle)

The system requires information related with zones' distribution and physical connections among zones (i.e., zone transition information). Therefore, it is necessary to have coarse-grained information about the area of interest. We define 14 zones in our environment. Each zone is a wall separated area (i.e., rooms, corridor).

In the zone prediction method, we setup three conceptually different machine learning algorithms (KNN, Multilayer Perceptron (MLP), and CART). The Python Scikit-learn library [51] was used to implement the individual machine learning algorithms. To build the zone fingerprinting database, we collected 9800 fingerprint instances, approximately 700 in each zone. The structure of a fingerprint instance consists of Wi-Fi and UWB RSS readings. Zone fingerprint database entries were collected equally distributed over the whole area in each zone. The data collection rate is only constrained by computational capabilities of the Wi-Fi sensor of the MT. Thus, in our experiments, every fingerprinting entry was collected at a rate of 3 entries per second. Since our approach does not need to predefine any survey point, the time needed to build the fingerprinting database is proportional to the number of collected instances multiplied by the instance collection rate. As introduced in Chapter 4, hyperparameters have significant impact on the performance of the machine learning algorithm. Thus, similar to Chapter 4, we use a nested cross validation technique to

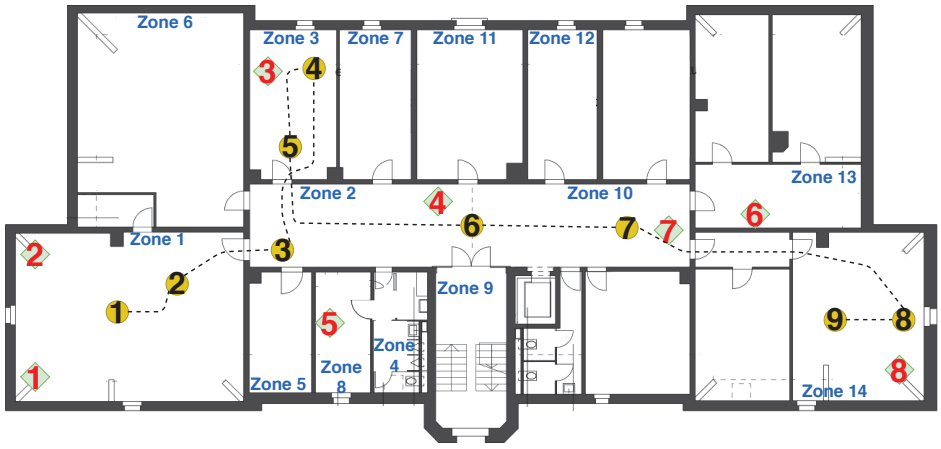


Figure 6.7: Cloud layer architecture

Figure 6.8: Scenario 1. Trajectory (black dotted lines), UWB-AN distribution (diamond green points), check points (yellow circles), and zone definition.

adjust them [76]. Thus, we configured hyperparameters to single hidden layer with 14 neurons for the MLP method. Activation function was configured to the sigmoid function [110]. For the CART method, we configure *gini* as function to measure the quality of a split [12]. The number of neighbors to use for classification was configured to 5 for the KNN method. Finally, to reduce the negative impact of environmental changes and different hardware, we use differential Wi-Fi RSS instead of absolute raw values.

6.4 Performance Evaluation

6.4.1 Measurement Setup

We evaluated the performance of our system in two office-like indoor scenarios along complex trajectories. Experiments were conducted on the third floor of the building of the Institute of Computer Science at the University of Bern. The experiments were designed to determine the parameter configuration that leads to the best performance of the system. Thus, we varied the number of particles and the number of UWB-ANs as configuration parameters.

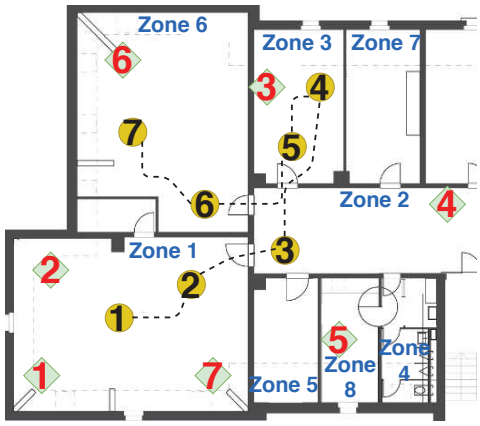


Figure 6.9: Cloud layer architecture

Figure 6.10: Scenario 2. Trajectory (black dotted lines), UWB-AN distribution (diamond green points), check points (yellow circles), and zone definition.

6.4.2 Performance Evaluation

In the first scenario, we deployed 8 UWB-ANs in an area of $702m^2$ ($39m$ x $18m$). In the second scenario, we increased the UWB-AN density by deploying 7 UWB-ANs in an area of $342m^2$ ($19m$ x $18m$). Several check points are defined along each trajectory to determine the tracking localization error. In the trajectory of scenario 1, we defined 9 check points, whereas 7 check points are defined along the trajectory in scenario 2. Distribution of the check points can be seen in Figures 6.7 and 6.9. Experiments were repeated twice. Therefore, 32 check points were analyzed. We define the tracking error as the Euclidian distance between the position calculated by the system and the ground truth position in of each check point. We compare our indoor tracking (InTrack) approach to the commercial solution Sequitur InGPS Lite [106]. Hereafter, we will refer to Sequitur InGPS as the commercial approach. The commercial approach was deployed for these experiments as part of a bachelor thesis project [111].

Figures 6.11 and 6.13 shows the cumulative distribution function (CDF) of localization error for the tracking algorithms in scenario 1 and scenario 2 respectively. In scenario 1, InTrack and the commercial tracking system show similar localization performance when the number of particles in InTrack is configured to 1500 and 1000. However, as shown in Figure 6.11, InTrack achieves higher accuracy and more stable performance compared to the commercial system when the number of particles is configured to

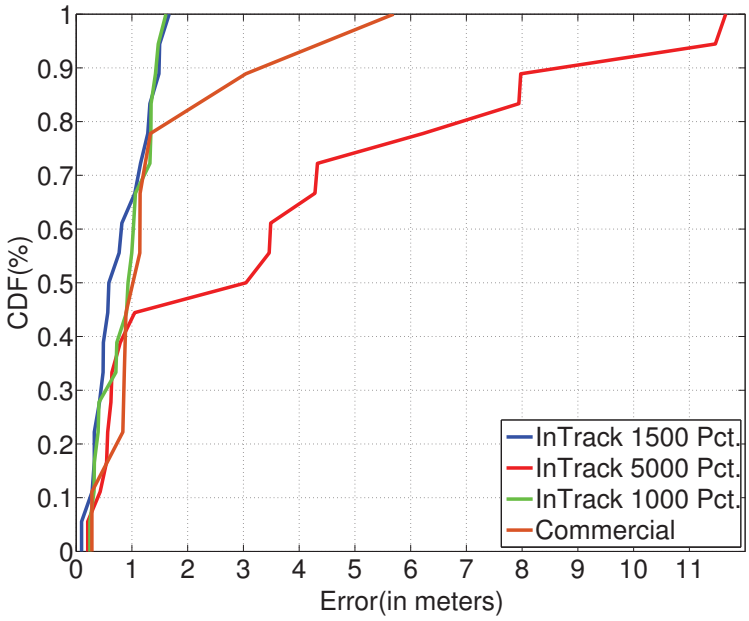


Figure 6.11: Empirical CDF of tracking error in Scenario 1.

1500 and 1000. However, the performance of InTrack decreases when the number of particles is set to 5000. This is because by increasing the number of particles, the processing time and the algorithm complexity also increase. Thus, it is clear that the performance of the tracking algorithm is detrimentally affected by processing time. Table 6.2 summarizes the average of tracking errors, standard deviation and 90% accuracy.

In scenario 1, InTrack achieves around $1.4m$ for 90% accuracy when the number of particles is set to 1000 and 1500. The mean error and standard deviation are also similar for these two configurations. However, the mean error is lower when the number of particles is set to 1000. Details about the tracking performance can be seen in Figures 6.12 and 6.1 which depict the mean error, standard deviation and 90% accuracy of the tracking errors in scenario 1 and scenario 2 respectively.

Since the localization algorithms are running on the Edge layer, InTrack's performance depends on the quality of the communication link between client devices and Edge servers. The commercial system uses UWB-based communication links to transmit data. Therefore, a low quality UWB link connection increases the tracking errors in the commercial system. This behaviour is evident in scenarios with low UWB-AN

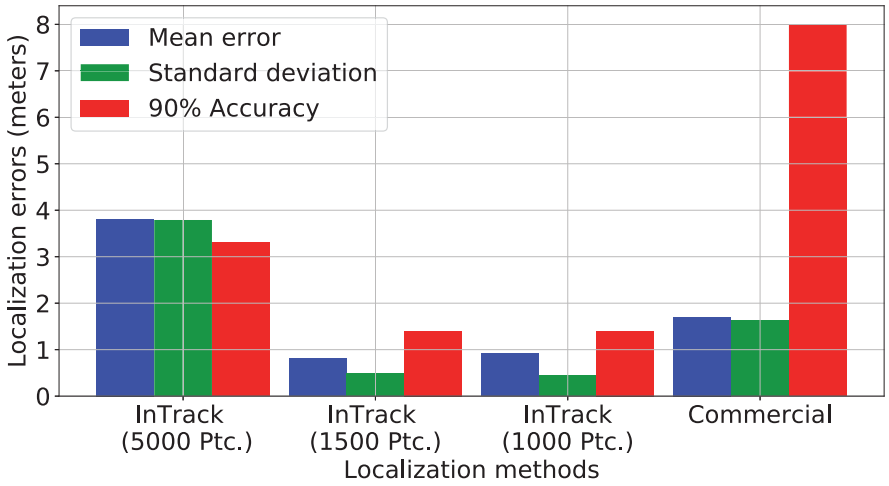


Figure 6.12: Tracking performance of the tracking algorithm in Scenario 1.

Table 6.2: Tracking performance summary.

Scenario	Tracking system	Mean error	S.D	90% Acc.
1	InTrack (5000 Ptc.)	3.81m	3.79m	3.3m
	InTrack (1500 Ptc.)	0.81m	0.48m	1.4m
	InTrack (1000 Ptc.)	0.92m	0.44m	1.4m
	Commercial	1.7m	1.63m	8.0m
2	InTrack (1500 Ptc.)	0.61m	0.23m	0.83m
	InTrack (1000 Ptc.)	0.52m	0.24m	0.83m
	Commercial	0.76m	0.15m	0.85m

density (i.e., low amount of UWB-AN in a large area of interest), such as scenario 1, where we observe data UWB communication problems due to long transmission distances. Unlike the commercial approach, InTrack implements a Websocket-based data communication module. Thus, InTrack overcomes the transmission problems observed in the commercial approach.

In scenario 2, InTrack and the commercial system show similar performance. However, as it can be seen in Figure 6.13 and Table 6.2, InTrack overcomes the commercial and the client-based systems. We increased the UWB-AN density to 8 UWB-ANs in an area of $702m^2$ and to 7 UWB-ANs in an area of $342m^2$. Moreover, the quality in the UWB communication link between the MD and the Edge server was improved by using Websocket technology. As shown in Table 6.2, the localization error in each check

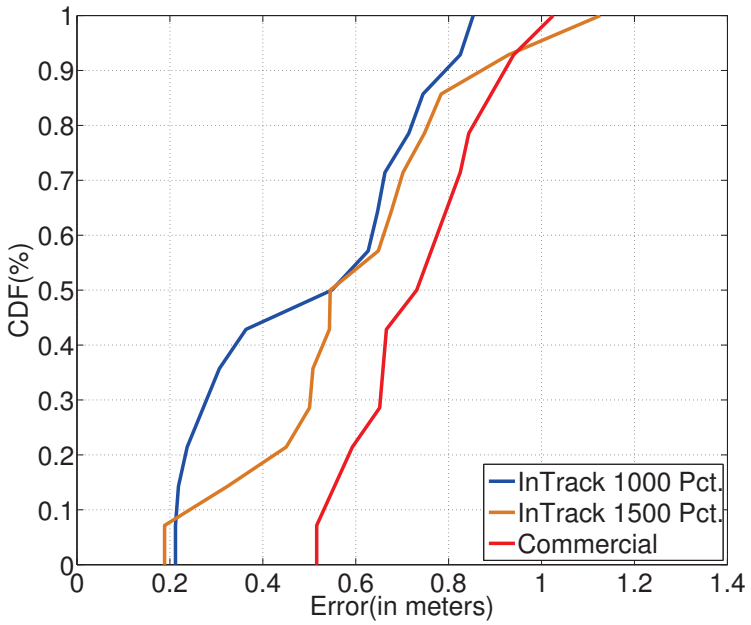


Figure 6.13: Empirical CDF of tracking error in Scenario 2.

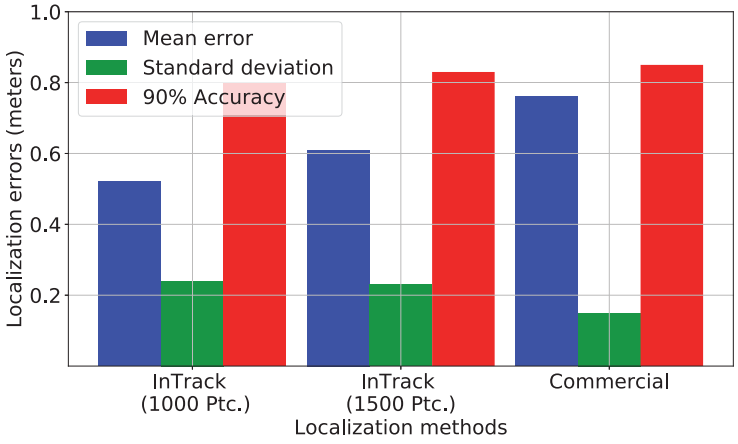


Figure 6.14: Tracking performance of the tracking algorithm in Scenario 2.

point is significantly reduced compared to scenario 1. It proves the importance of the UWB-AN density in the localization performance.

Even though the heavy computations (i.e., localization algorithms) are offloaded

from the client device to one Edge server, a setting of 5000 particles increases the processing time complexity of the tracking algorithm (see Figure 6.15). This affects the performance of the system. This proves that in real-time localization applications, processing time influences the accuracy performance of the localization system. Thus, we can observe in Figure 6.11 and Table 6.2 that InTrack with 1000 particles overcomes InTrack with 5000 particles by 75.8%, 88.4%, and 57.6% considering mean tracking error, standard deviation and 90% accuracy respectively.

6.4.3 System Complexity Analysis

In this subsection, we discuss the theoretical time complexity of our system, which mainly comes from the ensemble learning models for zone prediction and particle filter approach for data fusion. Thus, the theoretical time complexity of the system is the prediction time complexity of the zone prediction method plus the time complexity of the particle filter method. The prediction time complexity of the ensemble learning model is discussed in Chapter 4. Therefore, in the following paragraphs, we discuss the time complexity of the particle filter approach for data fusion.

The proposed particle filters method complexity is evaluated for the indoor tracking problem. In this subsection, we discuss the time complexity of the proposed particle filter for indoor tracking method from a theoretical point of view. The complexity of a computational algorithm can be estimated by counting the number of floating-point operations (FO) [55]. A FO is defined as subtraction, addition multiplication, or division of two floating-point numbers. However, there are some operations such as random number generation and non-linear function evaluation that cannot be estimated in terms of FOs. However, it is still possible to analyze the complexity by measuring the absolute computation time that the algorithm requires to complete its execution. Afterwards, a theoretical estimation of the complexity can be obtained by comparing the absolute computation time values to the number of FOs of the algorithm.

To estimate the time complexity, we analyse each process of the particle filter individually. Thus, the particle filter method is split into three processes: The Prediction, Observation and the Evaluation process. Afterwards, each process is split into its most representative sub-process. We estimate the number of FO and non-FO operations in each sub-process. The number of FO and non-FO operations is defined in terms of n and m, where n is the number of particles used in the particle filter and m is the

Table 6.3: Number of FO and non-FO operations for PF.

Process	Sub-process	FO	Non-FO
Prediction	Distribute particles	m	$2 \cdot m$
Observation	Compute ranging likelihood	$n \cdot m$	0
	Compute zone likelihood	n	0
	Compute particle's weight	$2 \cdot n$	0
Evaluation	Compute Neff	$2 \cdot n + 1$	0
	Systematic resampling	$3 \cdot n$	n
	Compute position	n	0

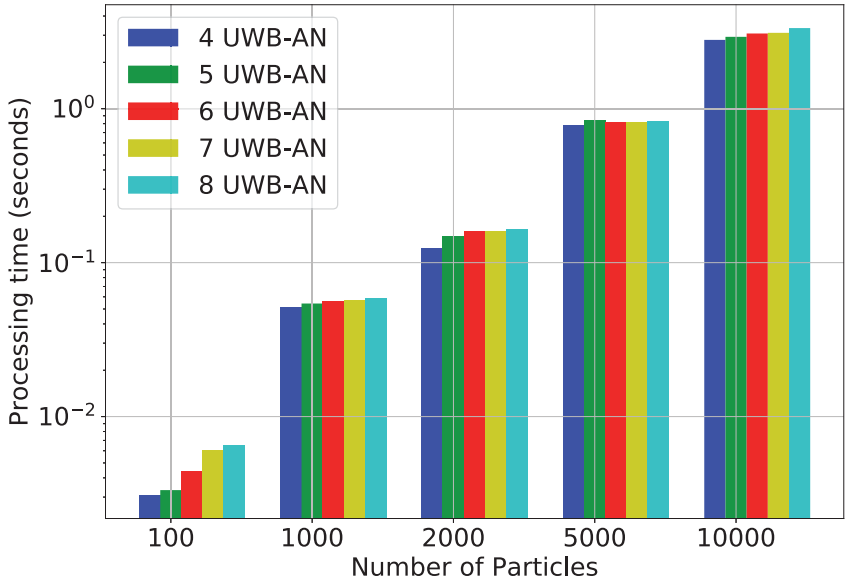


Figure 6.15: Processing time in terms of the number of particles and the number of UWB-AN

number of UWB-AN used for ranging.

Table 6.3 shows the number of FO and non-FO operations defined in the proposed particle filter method. In the Prediction process, particles are updated based on Equation 2.39. Thus, m FO and $2 \cdot m$ non-FO operations are performed. In the Observation process, calculating the ranging likelihood performs $n * m$ FOs, which is the most complex sub-process. It is because the ranging likelihood is computed for each particle by considering every UWB-AN. Therefore, the ranging likelihood

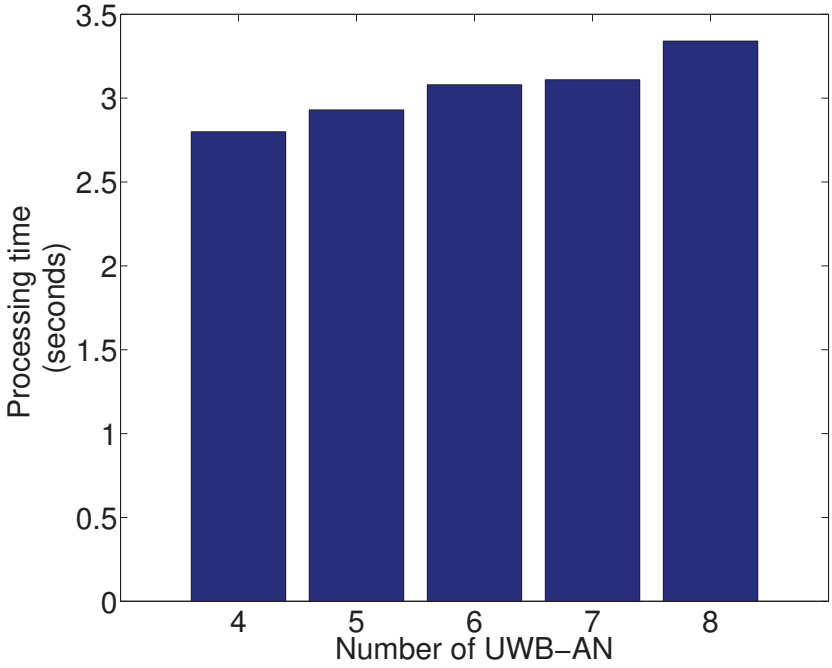


Figure 6.16: InTrack processing time regarding the number of UWB-AN

process performs a nested loop to iterate over each particle and each UWB-AN. In the Evaluation process, the systematic resampling is also a computational expensive sub-process. This is because particles with low weight are replaced by particles with high weight based on a CDF function. Thus, the systematic resampling sub-process iterates three times through the particles array. The number of operations of the particle filter method can be computed in terms of n and m as $O(n,m) = (n \cdot m) + 3m + 10 \cdot n + 1$ operations. Thus, considering the *Big O* notation to classify the performance characteristics of an algorithm, the degree of time complexity of our proposed particle filter method is $O(n \cdot m)$. Figure 6.15 depicts the processing time in terms of the n and m that are used in the tracking process. The time complexity of the particle filter method shows an exponential growth with respect to the number of particles. The number of particles increments faster than the number of UWB-AN. Therefore, the exponential growth of the processing time regarding n increments also faster than processing time regarding m (see Figure 6.15). Therefore, when fusing in a particle filter zone prediction results and radio ranging information, the time complexity of the localization system grows exponentially regarding the number of extracted features, number of training samples, number of particles, and number of

ANs used in the localization process.

Figure 6.16 depicts processing time regarding m when n is set to 10000. It can be seen that the processing time also grows exponentially regarding m. Thus, observations depicted in Figures 6.16 and 6.15 confirms the theoretical complexity analysis presented in Table 6.3.

6.5 Conclusions

In this chapter, we presented a mobile edge computing-based system for indoor tracking of IoT mobile devices. The system is working based on the fusion of embedded IMU readings, WiFi RSSI readings, floor plan information, and UWB radio signals. We proposed an enhanced particle filter to fuse different types of information. Moreover, in this chapter we provided a deep performance analysis of a mobile edge computing-based architecture for an indoor tracking system, which offloads the resource-demanding processing tasks to edge servers that are close to end users. We analyzed the impact of the number of particles and the number of ANs on the tracking performance. This highlights the influence of the processing time complexity over the tracking performance. Moreover, we performed experiments in office-like environments using different system parameters to maximize the tracking performance. Experiment results show that InTrack achieves an average tracking error of $0.52m$ and 90% accuracy is $0.83m$. It outperforms some commercial products and client-based tracking systems. Thus, by bringing cloud computing capabilities to the network edge, InTrack is more accurate and robust than traditional and commercial indoor localization methods.

7

Conclusions and Outlook

We first summarize the contributions of this thesis in the order of their occurrence in the thesis in Section 7.1. Afterwards, we briefly discuss possible future directions in the field of indoor tracking in Section 7.2. Our first contribution is enhanced solutions for improving indoor localization and tracking performance. We presented an enhanced particle filter to fuse radio range information, IMUs as well as floor plan information for indoor tracking. Moreover, we proposed an asynchronous continuous correction phase to mitigate the tracking errors caused by unstable sensors readings on commodity mobile devices. The second contribution is localization with failure recovery. We proposed an efficient method to recover the localization system from localization failures. Since the recovery method relies on zone level localization results, we also focused on providing enhanced methods to achieve high zone level localization accuracy. The third contribution is the localization system architectures. We proposed distributed system architectures where lightweight algorithms run on the mobile target devices, whereas heavy calculations are offloaded to nearby edge servers. Thus, algorithmic complexity is not constrained by the limited computational resources of target mobiles devices. Thus, we extended our localization approaches to

a distributed system architecture.

7.1 Main Contributions

In this section, we summarize the contributions of this thesis. The contributions are presented in the order of occurrence in the thesis.

In Chapter 3, our tracking algorithm exploits an enhanced particle filter approach to fuse radio signals, inertial sensors and physical information of the environment. Additionally, we presented an asynchronous continuous correction phase that is able to tackle the low sampling rate problem of Wi-Fi sensors on the smartphone side. Furthermore, we presented an enhanced PDR method to further improve the heading orientation estimation. The localization approach is formulated in a discretized graph-based representation of the indoor environment. We evaluated our localization system in complex large trajectories in different indoor scenarios. Localization experiments show that our approach can achieve an average tracking error of $1.01m$ and 90% accuracy is $1.7m$. Moreover, we have shown that processing time grows exponentially when increasing the number of particles. We demonstrated that high processing time leads to decrease the localization performance of the system.

In Chapter 4, we focused on localization failure recovery methods to achieve robust localization performance. We presented a simple method to recover the system from localization failures. The proposed failure recovery method relies on machine learning approaches to provide zone level localization. Thus, we also focused on enhanced machine learning methods to provide zone level localization. We proposed two novel ensemble learning algorithms. The first proposed ensemble learning algorithm (COND) is based on the concept of conditional probabilities. We considered the prediction performance of individual learning methods to improve the prediction accuracy in an ensemble learning model. We have validated the performance of the prediction model by using different smartphone sensor measurements, such as Wi-Fi RSS, MF readings. Evaluation results show that our proposed ensemble predictor COND achieves room prediction accuracy of 96.8%. Although, the prediction performance of COND overcomes its base machine learning methods and SV, the prediction time complexity of COND is higher than its base machine learning methods. When using 17569 training samples and 10 extracted features, the prediction time complexity is $O(18x10^4)$. However, the prediction time complexity of COND grows

exponentially with the number of extracted features and the number of training samples. This exponentially growing will lead to increase execution time when using bigger training databases.

In Chapter 4, we also introduced HMM-d for zone level localization. HMM-d integrates information about transition probabilities between zones and discriminative learning methods in a Hidden Markov Model. Thus, we presented a probabilistic-based system to achieve high zone level localizaion accuracy. We evaluated the HMM-d predictor in a complex real-world indoor environment. Evaluation results indicated that HMM-d approach is more accurate and robust than individual learning and voting-based models. Moreover, we evaluated the average prediction processing time of the HMM-d method. Results showed that processing time of the HMM-d method is lower than the processing time of the Soft Voting method. This means that processing a Hidden Markov Model takes lower computational efforts than processing the soft voting rule. However, the prediction time complexity of HMM-d is higher than its base predictors. When using 7 extracted features and 10000 training samples, the prediction time complexity of HMM-d is $O(70x10^3)$. However, the prediction time complexity of the HMM-d method grows exponentially with the number of extracted features and the number of training samples. Thus, appropriate processing resources to run the HMM-d method are the underlying requirement to guarantee low prediction execution time.

Mobile devices need to deal with limited battery and limited processing resources, especially for interactive resource-intensive mobile applications such as indoor localization. In Chapters 5 and 6, we proposed distributed architectures for indoor tracking. The lightweight methods run on the mobile target device and heavy computations are offloaded to a third part server. Thus, by exploiting distributed system architectures, we focus on enhanced methods to provide reliable and accurate indoor tracking.

In Chapter 5, we investigated how to improve the resampling process to assure a particle distribution over areas where the desired distribution is large. We proposed a particle filter-based reinforcement learning (PFRL) approach for autonomous robust wireless indoor positioning for smartphones. A reinforcement learning method is applied on top of the proposed particle filter to improve the system robustness against localization failures. Moreover, the reinforcement learning method makes the localization system converge much faster than other particle-based localization methods. The execution time complexity of the reinforcement learning method grows

linearly regarding the number of states and number of actions. Thus, the execution time complexity of the reinforcement learning method does not increase the overall time complexity of the localization system, which grows exponentially regarding the number of extracted features, number of training samples, number of particles, and number of ANs used in the localization process.

PFRL was implemented on a distributed machine learning-based network architecture. To validate environmental independence of the PFRL method, we evaluated PFRL in two complex real-world indoor scenarios. Moreover, we compared the PFRL localization performance to a client-based PFRL approach. Results have shown that the localization performance of PFRL overcomes the localization performance of the client-based approach. This demonstrated the negative influence produced by limited processing resources of mobile devices when increasing the processing time complexity of the localization algorithms.

In Chapter 6, we proposed a three layer MEC-based system architecture to provide indoor localization and tracking services to mobile devices in the IoT domain. By exploiting MEC architecture, we extended the localization approach presented in Chapter 5. We included a cloud layer to allow remote access monitoring to IoT devices information. Moreover, we provided a probabilistic method to integrate machine learning zone prediction results to radio frequency and environmental information for accurate indoor localization. Since, the computational complexity is offloaded to a MEC server, the localization algorithms are not constrained by the limited computational resources of mobile devices. We discussed the computational complexity of the proposed localization system. When fusing in a particle filter zone prediction results and radio ranging information, the time complexity of the localization system grows exponentially regarding the number of extracted features, number of training samples, number of particles, and number of ANs used in the localization process. Additionally, we tested the localization performance of the MEC-based localization system in complex office-like scenarios. Experiments results show that our approach can achieve an average tracking error of $0.44m$ and 90% accuracy is $0.6m$. It outperforms some commercial products and terminal-based tracking systems.

7.2 Future Directions

In this thesis, we provide enhanced solutions to improve indoor tracking and localization performance. We validated our proposed solutions in real complex indoor scenarios. While localization approaches presented in this thesis are significant contribution for the deployment of indoor localization systems, other opportunities for extending the scope of this thesis remain. In the following paragraphs, we briefly discuss some of these opportunities to extend the work presented in this thesis.

Propagation of indoor radio signals is affected by the environmental layout, such as the number of walls, floors, furniture density, and mobility. This radio propagation instability affects localization accuracy especially in fingerprinting localization approaches. To reduce the negative impact of environmental dependency, we use differential Wi-Fi RSS instead of absolute raw values [108]. However, advanced solutions to deal with this environmental dependency of radio signal propagation is still an open research problem. Enhanced machine learning approaches, such as neural networks can be used to learn radio signals propagation patterns. Thus, machine learning based signal propagation models can be used to deal with the negative influence of radio propagation instability in indoor localization systems.

PDR systems use embedded sensors of mobile devices to derive pedestrian relative movements. PDR systems usually consist of three stages: step detection, stride length estimation, and heading direction estimation. However, there are still some limitations in the existing PDR techniques. PDR methods presented in thesis assume that the heading direction offset and stride length remains constant. The heading direction angle offset is the angle between the direction of smartphone and the direction of pedestrian. Moreover, the step detection method relies on simple linear acceleration threshold. The assumption regarding the heading direction offset angle is met when pedestrians handhold the smartphones on the front of the body. However, in real scenarios, the smartphone attitude is arbitrary. Thus, neither the heading direction offset and stride length can be guaranteed to be constant. Therefore, further PDR solutions must be provided. Heading direction offset assumption can be handled by representing rotations based on quaternion concepts [80]. Whereas, there are several works aimed to provide adaptive stride estimation and enhanced step detection methods to further improve PDR systems [49] [96].

8

List of Acronyms

ACK Acknowledgment Message

AN Anchor Node

AP Access Point

CC Cloud Computing

CFS Customer Facing Service

CNN Convolutional Neural Network

COND Conditional Classifier

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CoAP The Constrained Application Protocol

ED Equally Distributed

FCS Frame Check Sequence

FP False Positive

FN False Negative

FO Floating-point Operation

GNSS Global Navigation Satellite System

GPS Global Positioning System

HDE Heuristic Drift Elimination

HMM HIdden Markov Model

HMM-d HMM-discriminatine Ensemble Learning Method

IoT Internet of Things

IMU Inertial Measurement Unit

IS Importance Sampling

ITS Intelligent Transportation System

KF Kalman Filter

KNN K-Nearest Neighbors

KRP Kidnapping Robot Problem

LA Learning Agent

LBS Location-Based Service

LDPL Log Distance Path Loss

LOS Line of Sight

MAC Medium Access Control

MCMC Markov Chain Monte Carlo

MCL Monte Carlo Localization

MCLS Monte Carlo Localization System

ME Mobile Edge

MEC Multi-Access Edge Computing

MEO Multi-Access Edge Orchestrator

MF Magnetic Field

MIMO Multiple Input Multiple Output

ML Machine Learning

MLP Multi Layer Perceptron

MT Mobile Target

MQTT Message Queuing Telemetry Transpor

M2M Machien to Machine

NB Naive Bayes

NLOS Non Line of Sight

NLR Nonlinear Regression Model

PCF Point Coordination Function

PDF Posterior Distribution Function

PDR Pedestrian Dead Reckogning

PF Particle Filters

PFRL Particle Filter-based Reinforcement Learning

PHY Physical Layer

POI Point of Interest

RAN Radio Access Network

RFID Radio Frequency

RSSI Received Signal Strength Information

RF Radio Frequency

RL Reinforcement Learning

SLAM Simultaneous Localization and Mapping

SMC Sequential Monte Carlo

SVM Support Vector Machine

TOF Time of Flight

TP True Positive

QoE Quality of Experience

QoS Quality of Service

UE User Equipment

UWB Ultra Wide Band

UWB-AN Ultra Wide Band Anchor Node

WAN Wireless Local Area Network

Bibliography

[1] I. 80211n 2009. (2009) Ieee standard for information technology-local and metropolitan area networks-specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy specifications amendment 5: Enhancements for higher throughput. [Online]. Available: https://standards.ieee.org/findstds/standard/802.11n-2009.html

[2] I. 802.15.4-2015. (2016) 802.15.4-2015 - ieee standard for low-rate wireless networks. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7460875

[3] J. L. A. Mariakakis, S. Sen and K. Kim., "Sail: Single access point-bases indoor localization." *Proceedings of the 12th annual conference on Mobile systems, applications, and services MobiSys 14*, vol. 14, pp. 315–328, June 2014.

[4] H. Abdelnasser, R. Mohamed, A. Elgohary, M. Alzantot, H. Wang, S. Sen, R. Choudhury, and M. Youssef, "Semanticslam: Using environment landmarks for unsupervised indoor localization." *IEEE Transactions on Mobile Computing*, vol. 15, pp. 1770–1782, 2016.

[5] Android Documentation, "2016. Android Position Sensorsiii. Retrieved Jun 7, 2016 from developer.android.com/guide/topics/sensors/."

[6] M. Bachtler, "Kalman filter supported wifi and pdr based indoor positioning system bachelor thesis," Bachelor Thesis, University of Bern, Institute of Computer Science, February 2018.

[7] P. Bahl and V. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," *INFOCOM 2000. Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, no. 19, 2000.

[8] M. Beck and M. Maier, "Mobile edge computing: Challenges for future virtual network embedding algorithms," *ADVCOMP 2014 : The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 65–70, September 2014.

[9] R. Bellman, *The Theory of Dynamic Programming*. 1700 Main street, Santa Monica, California, USA: The RAND Corporation, 1954.

[10] J. Borestein and L. Ojeda., "Heuristic drift elimination for personnel tracking systems." *The Journal of Navigation*, vol. 63, no. 4, pp. 591–606, November 2010.

[11] O. Bousquet, U. von Luxburg, and G. Ratsch, "Bayesian inference: An introduction to principles and practice in machine learning." *Fresenius Environmental Bulletin*, vol. 20, no. 5, 2004.

[12] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "Api design for machine learning software: experiences from the scikit-learn project," *PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, November 2013.

[13] B. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Elsevier Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, December 2008.

[14] J. Carrera, Z. Zhao, and T. Braun, "Conditional probability-based ensemble learning for indoor landmark localization," *Elsevier Computer Communications*.

[15] ——, "Mec-based uwb indoor tracking systemn," *IEEE/IFIP Wireless On-demand Network systems and Services Conference (WONS 2019)*, vol. 15, January 2019.

[16] J. Carrera, Z. Zhao, T. Braun, and Z. Li, "A particel filter-based reinforcement learning approach for reliable wireless indoor positioning (accepted for publication)," *IEEE JSAC Issue on Machine Learning in Wireless Communications, 2019*, July 2019.

[17] J. Carrera, Z. Zhao, T. Braun, Z. Li, and A. Neto, "A real-time robust indoor tracking system in smartphones," *Elsevier Computer Communications*, vol. 117, pp. 104–115, February 2018.

[18] J. Carrera, Z. Zhao, T. Braun, H. Luo, and F. Zhao. (2018) Discriminative learning-based smartphone indoor localization (submitted). [Online]. Available: http://arxiv.org/abs/1804.03961

[19] J. Carrera, Z. Li, Z. Zhao, and T. Braun, "A real-time indoor tracking system by fusing inertial sensor, radio signal and floor plan," *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, October 2016.

[20] J. Carrera, Z. Li, Z. Zhao, T. Braun, and A. Neto, "A real-time indoor tracking system in smartphones," *MSWiM '16 Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 292–301, November 2016.

[21] J. Carrera, Z. Zhao, and T. Braun, "Room recognition using discriminative ensemble learning with hidden markov models for smartphones," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC)*, September 2018.

[22] M. Centenaro, L. V. A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios," *IEEE Wireless Communications*, vol. 23, pp. 60–67, 2016.

[23] S. Central. (2019) What is edge computing, multi-access edge computing (mec)? [Online]. Available: https://www.sdxcentral.com/edge/definitions/what-multi-access-edge-computing-mec/

[24] X. Chai and Q. Yang, "Reducing the calibration effort for probabilistic indoor location estimation." *IEEE Tran. Mobile Computing.*, vol. 6, no. 6, pp. 649–662, July 2016.

[25] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349 – 359, July 2014.

[26] G. Cleary and E. Trigg, "K*: An instance-based learner using an entropic distance measure," in *12th International Conference on Machine Learning*, 1995, pp. 108–114.

[27] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan, "How close is close enough? understanding the role of cloudlets in supporting display

appropriation by mobile users," in *Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communication (PerCom)*. United States: Institute of Electrical and Electronics Engineers, 2012, pp. 122–127.

[28] F. G. D. Hol, B. Schon. (2018) On resampling algorithms for particle filters. [Online]. Available: http://people.isy.liu.se/rt/schon/Publications/HolSG2006.pdf

[29] F. Darbari, R. Stewart, I. MaGregor, G. Whyte, and I. Thayne, "Channel estimation for short range wireless sensor network," *EURASIP*, 2005.

[30] S. David, "An introduction to particle filters for tracking and guidance," *Advances in Missile Guidance, Control, and Estimation*, pp. 437–476, 09 2012.

[31] F. Dieter, w. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *J. Artif. Int. Res.*, vol. 11, no. 1, pp. 391–427, Jul. 1999.

[32] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[33] S. Electronics, "Dcm tutorial-an introduction to orientation kinematics," http://www.starlino.com/dcmtutorial.html, 2011.

[34] ETSI. (2016) Mobile edge computing (mec); framework and reference architecture. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/mec/001_099/003/01.01.01_60/gs_mec003v010101p.pdf

[35] ——. (2017) European telecommunications standards institute, mobile edge computing portal.

[36] L. W. Y. F. F. Hong, H. Chu and Z. Guo., "Pocket mattering: Indoor pedestrian tracking with commercial smartphone," *International Conference on Indoor Positioning and Indoor Navigation*, November 2012.

[37] B. Ferris, D. Hahnel, and D. Fox, "Gaussian processes for signal strength-based location estimation." *Procedures of Robotics Science and Systems*, 2006.

[38] I. Fette and A. Melnikov, "The websocket protocol," *Internet Engineering Task Force (IETF)*, 2011.

[39] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, March 1973.

[40] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[41] Z. Ghahramani, "An introduction to hidden markov models and bayesian networks." *International Journal of Pattern Recognition and Artificial Intelligence*, 2001.

[42] R. Goodrich. (2013) Accelerometers: What they are and how they work. [Online]. Available: https://www.livescience.com/40102-accelerometers.html

[43] K. Guan, L. Ma, X. Tan, and S. Guo, "Vision-based indoor localization approach based on surf and landmark," *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2016.

[44] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, pp. 6–23, Jan 1997.

[45] S. Hay and R. Havinga, "On the calibration and performance of rss-based localization methods," *IEEE Internet of Things (IOT)*, 2010.

[46] S. He and S. Chan, "Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons." *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 466 – 490, 2016.

[47] S. He, S. G. Chan, L. Yu, and N. Liu, "Calibration-free fusion of step counter and wireless fingerprints for indoor localization," *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2015)*, 2014.

[48] S. Hilsenbeck, D. Bobkov, G. Schroth, R. Huilt, and E. Steinbach, "Graph-based data fusion of pedometer and wifi measurements for mobile indoor positioning," *The 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2014)*, September 2014.

[49] N.-H. Ho, P. Truong, and G.-M. Jeong, "Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone," *Sensors*, vol. 16, p. 1423, 09 2016.

[50] F. Hong, Y. Zhang, M. Wei, Y. Feng, and Z. Guo, "Wap: Indoor localization and tracking using wifi-assited particle filter," *Annual IEEE Conference on Local Computer Networks LCN*, no. 39, pp. 210–217, June 2014.

[51] INRIA. (2018) Scikit-learn. machine learning in python. [Online]. Available: http://scikit-learn.org/stable/

[52] j. Marsan. (2018) Weka-for-android. [Online]. Available: https://github.com/rjmarsan/Weka-for-Android

[53] J. Jung, S. Lee, and H. Myung, "Indoor mobile robot localization and mapping based on ambient magnetic fields and aiding radio sources," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, November 2014.

[54] N. Kakiuchi and S. Kamijo., "Pedestrian dead reckoning for mobile phones trhough walking and running mode recognition." *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013)*, vol. 63, pp. 261–267, October 2013.

[55] R. Karlsson, T. Schon, and F. Gustafsson, "Complexity analysis of the marginalized particle filter," *EDICS: 2-ADPT Adaptive Systems and Filtering*, 2004.

[56] G. Kitagawa, "Monte carlo filter and smoother and non-gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, pp. 1–25, 1996.

[57] R. Kálmán, "A new approach to linear filtering and predition problems," *Journal of Basic Engineering*, pp. 35–45, January 1960.

[58] B. Lakmali, W. Wijesinghe, K. de SIva, K. Liyanagama, and S. Dias, "Design, implementation & testing of positioning techniques in mobile networks," *The 3rd International Conference on Information and Automation for Sustainability*, 2007.

[59] Z. Li, T. Braun, and D. Dimitrova, "A passive wifi source localization system based on fine-grained power-based trilateration," *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015.

[60] Z. Li, T. Braun, X. Zhao, and Z. Zhao, "A narrow-band indoor positioning system by fusing time and received signal strength via ensemble learning," *IEEE Access 2018*, pp. 9936–9950, 2018.

[61] Z. Li, D. Burbano, Z. Zhao, J. Carrera, and T. Braun, "Fine-grained indoor tracking by fusing inertial sensor and physical layer information in wlans," *IEEE International Conference on Communications (ICC)*, 2016.

[62] Z. Li, D., and T. Braun, "Passively track wifi users with an enhanced particle filter using power-based ranging," *IEEE Transactions on Wireless Communications*, vol. 16, pp. 4599–4605, Nov 2017.

[63] Z. Li. (2016) Fine-grained indoor positioning and tracking systems. [Online]. Available: http://rvs.unibe.ch/research/pub_files/Li16.pdf

[64] H. Linde, "On aspects of indoor localization," PhD dissertation, University at Dortmund, 2006.

[65] J. Liu and R. Chen, "Sequential monte carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, 1998.

[66] J. Liu, L. P. R. Chen, R. Guinness, and H. Kuusniemi, "A hybrid smartphone indoor positioning solution for mobile lbs," *Sensors*, 2012.

[67] S. Liu, H. Luo, and S. Zou, "A low-cost and accurate indoor localization algorithm using label propagation based semi supervised learning." *Fifth International Conference Mobile Ad-Hoc and Sensor Networks*, pp. 108–111, 2009.

[68] D. Madigan, E. Einahrawy, R. Martin, W. Ju, P. krishnan, and A. Krishnakumar, "Bayesian indoor positioning systems." *IEEE. INFOCOM*, pp. 1217–1227, 2005.

[69] D. Mascharka and E. Manley, "Lips: Learning based indoor positioning system using mobile phone-based sensors." *2016 13th IEEE Annual Consumer Communications and Networking Conference (CCNC)*, pp. 968–971, 2016.

[70] A. Masiero, A. Guarnieri, F. Pirotti, and A. Vettore, "A particle filter for smartphone-based indoor pedestrian navigation," www.mdpi.com/journal/micromachines, 2014, iSSN 2072-666X.

[71] F. Melo. (2018) Convergence of q-learning: a simple proof. [Online]. Available: http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf

[72] ——. (2018) Support vector machines. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[73] M. Michael, T. Sebastian, K. Daphne, and W. Ben, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 2003.

[74] E. Mubashir and H. Rehman, "Internet of things (iot): A vision, architectural elements, and future directions," *Elsevier Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, September 2013.

[75] ——, "Mobile edge computing: Opportunities, solutions, and challenges," *EEEE Network*, vol. 70, no. 2, pp. 59–63, May 2017.

[76] K.-R. Müller, M. Krauledat, G. Dornhege, G. Curio, and B. Blankertz, "Machine learning techniques for brain-computer interfaces." *BIOMEDICAL ENGINEERING*, pp. 11–22, 2004.

[77] P. Nagpal and R. Rashidzadeh, "Indoor positioning using magnetic compass and accelerometer of smartphones," *'International Conference on Selected Topics in Mobile and Wireless Networking MoWNet*, pp. 140–145, 2013.

[78] J. Niklaus, "Machine learning for indoor positioning," Bachelor Thesis, University of Bern, Institute of Computer Science, November 2017.

[79] OpenCV.org, "Understanding k-Nearest Neighbour, 2016 from http://docs.opencv.org/3.0-beta/doc/py-tutorials/py-ml/py-knn/py-knn-understanding/py-knn-understanding.html."

[80] Opengl. (2018) Quaternion implementation tutorial. [Online]. Available: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/

[81] R. W. Ouyang, A. K. S. Wong, C. T. Lea, and M. Chiang, "Indoor location estimation with reduced calibration exploiting unlabeled data via hybrid generative/discriminative learning." *IEEE Transactions on Mobile Computing*, vol. 11, pp. 1613 – 1626, November 2012.

[82] M. Pelka, D. Amann, M. Cimdins, and H. Hellbriick, "Evaluation of time-based ranging methods: Does the choice matter?." *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, January 2018.

[83] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[84] S. Richard and A. Barto, *Reinforcement Learning: an Introduction*. San Mateo, CA: The MIT Press, 2019.

[85] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter, "Analysis of the scalability of uwb indoor localization solutions for high user densities," *Sensors*, vol. 18, no. 6, 2018.

[86] I. Rotoview. (2013) Gyroscopes. [Online]. Available: https://www.rotoview.com/gyroscope.htm

[87] ——. (2013) Magnetometer in smartphones and tablets. [Online]. Available: https://www.rotoview.com/magnetometer.htm

[88] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture," *IEEE Consumer Electronics Magazine*, pp. 65–70, October 2016.

[89] B. Schilit, N. Adams, and R. Want. (2018) Context-aware computing applications. [Online]. Available: https://ieeexplore.ieee.org/document/4624429

[90] scikit learn.org. (2017, March) Multi layer perceptron. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[91] ——. (2017, March) sklearn.ensemble.votingclassifier. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

[92] ——. (2017, March) Soft voting classifier. [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier

[93] S. Seidel and T. Rappaport, "914 mhz path loss predictions models for indoor wireless communications in multifloored buildings." *IEEE Transactions on Antennas and Propagation*, 1992.

[94] Sewio. (2018) Uwb: Technology-two way ranging. [Online]. Available: https://www.sewio.net/uwb-technology/two-way-ranging/

[95] ——. (2018) Uwb: Technology-two way ranging. [Online]. Available: https://www.sewio.net/uwb-technology/two-way-ranging/

[96] S. H. Shin and C. G. Park, "Adaptive step length estimation algorithm using optimal parameters and movement status awareness," *Elseivier Medical Engineering and Physics*, vol. 33, pp. 1064–1071, 11 2011.

[97] Y. Shu, C. Bo, G. Shen, C. Zhao, L. Li, and F. Zhao, "Signalslam: Simultaneous localization and mapping with mixed wi-fi, bluetooth, lte and magnetic signals." *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2014.

[98] ——, "Magicol: Indoor localization using pervasive magnetic field and opportunistic wi-fi sensing." *IEEE Journal on Selected Areas in Communications*, 2015.

[99] M. Speekenbrink, "A tutorial on particle filters," *Journal of Mathematical Psychology*, vol. 73, pp. 140–152, 2016.

[100] C. Systems. (2018, Mar.) Securing the internet of things: A proposed framework. [Online]. Available: https://www.cisco.com/c/en/us/about/security-center/secure-iot-proposed-framework.html

[101] H. Thornburg. (2016, March) Accuracy, precision, recall and f1 score: Interpretation of performance measures. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[102] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," https://docs.ufpr.br/ danielsantos/ProbabilisticRobotics.pdf, 2000.

[103] T. Tornado-Authors. (2018) Tornado: Running and deploying. [Online]. Available: http://www.tornadoweb.org/en/stable/guide/running.html

[104] M. Tory and T. Moller, "Rethinking visualization: A high-level taxonomy," in *Proceedings of the IEEE Symposium on Information Visualization*, ser. INFOVIS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 151–158. [Online]. Available: http://dx.doi.org/10.1109/INFOVIS.2004.59

[105] X. Tuyen, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *EEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, April 2017.

[106] UNISET, "Uniset company indoor tracking solution." [Online]. Available: http://www.unisetcompany.com/sequitur-family-en/

[107] M. Wallbaum and O. Spaniol, "Indoor positioning using wireless local area networks," *in Proceedings of the IEEE John Vincent Atanasoff International Symposium on Modern Computing.*, pp. 17–26, Oct 2006.

[108] J. Wang, Q. Gao, H. Wang, H. Chen, and M. Jin, "Differential radio map-based robust indoor localization," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, p. 12, 2011.

[109] C. Watkins and P. Dayan, "Technical note q-learning," *Machine Learning Technical Note*, vol. 8, pp. 279–292, 1992.

[110] D. WEKA. (2018, Mar.) Class multilayerperceptron. [Online]. Available: http://weka.sourceforge.net/doc.dev/

[111] M. Wenger, "Indoor positioning using raspberry pi with uwb," Bachelor Thesis, University of Bern, Institute of Computer Science, January 2019.

[112] C. Wu, Z. Yang, Z. Zhou, K. Qian, Y. Yunhao, and M. Liu, "Phaseu: Real-time los identification with wifi," *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[113] K. Wu, J. Xiao, Y. Yi, M. Gao, and L. Ni, "Fila: Fine-grained indoor localization," *INFOCOM Proceedings IEEE*, pp. 2210–2218, 2012.

[114] H. Xie, T. Gu, X. Tao, H. Ye, and J. Lu, "A reliability-augmented particle filter for magnetic fingerprinting based indoor localization on smartphone." *IEEE Transactions on Mobile Computing*, 2016.

[115] H. Xu, Z. Yang, Z. Zhou, L. Shangguan, K. Yi, and Y. Liu, "Indoor localization via multi-modal sensing on smartphones," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 208–219. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971668

Declaration of consent

on the basis of Article 30 of the RSL Phil.-nat. 18

Name/First Name: Carrera Villacrés José Luis

Matriculation Number: 13-506-936

Study program: Computer Science

Bachelor ☐ Master ☐ Dissertation ☑

Title of the thesis: Indoor Positioning and Tracking Methods for Mobile Wireless Devices

Supervisor: Prof. Dr. Torsten Braun

I declare herewith that this thesis is my own work and that I have not used any sources other than those stated. I have indicated the adoption of quotations as well as thoughts taken from other authors as such in the thesis. I am aware that the Senate pursuant to Article 28. RSL Phil.-nat. 05 is authorized to revoke the title awarded on the basis of this thesis.
For the purposes of evaluation and verification of compliance with the declaration of originality and the regulations governing plagiarism, I hereby grant the University of Bern the right to process my personal data and to perform the acts of use this requires, in particular, to reproduce the written thesis and to store it permanently in a database, and to use said database, or to make said database available, to enable comparison with future theses submitted by others.

Bern, 27/08/2019

Place/Date

Signature

José Luis Carrera Villacrés

Curriculum Vitae

Rte. Bertigny 43
1700 Fribourg
Switzerland
+41 (79) 76 26766
jlcarvi@hotmail.com

Education

2015–2019 **PhD. Computer Science (finishing October 2019)**, *University of Bern*, Communication and Distributed Systems Research Group, *Bern - Switzerland*.

2013–2015 **Master of Science in Computer Science**, *University of Neuchâtel, Fribourg and Bern*, Switzerland, *Swiss Joint Master of Science in Computer Science*.

2013–2015 **Master of Science in Technology and Communication Systems Management**, *National Polytechnic School (EPN)*, Quito-Ecuador, .

2001–2005 **Engineer in Information and Computer Systems**, *National Polytechnic School (EPN)*, Quito - Ecuador.

Scientific Publications in this Thesis

J.L. Carrera Villacrés, Z. Zhao, , T. Braun, and Z. Li, "A particle filter-based reinforcement learning approach for reliable wireless indoor positioning," *IEEE JSAC Issue on Machine Learning in Wireless Communications*, 2019.

J.L. Carrera Villacrés, Z. Zhao, M. Wenger, and T. Braun, "Mec-based uwb indoor tracking systemn," *IEEE/IFIP Wireless On-demand Network systems and Services Conference (WONS 2019)*, vol. 15, January 2019.

J.L. Carrera Villacrés, Z. Zhao, T. Braun, and Z. Li, "Real-time smartphone indoor tracking using particle filter with ensemble learning methods," *IEEE Conference on Local Computer Networks (IEEE LCN), Chicago, USA.*, October 2018.

Z. Zhao, **J.L. Carrera Villacrés**, J. Niklaus, and T. Braun, "Conditional probability-based ensemble learning for indoor landmark localization," *Elsevier Computer Communications*, 2019.

J.L Carrera Villacrés, Z. Zhao, and T. Braun, "Room recognition using discriminative ensemble learning with hidden markov models for smartphones," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC)*, September 2018.

J.L Carrera Villacrés, Z. Zhao, T. Braun, Z. Li, and A. Neto, "A real-time robust indoor tracking system in smartphones," *Elsevier Computer Communications*, vol. 117, pp. 104–115, February 2018.

J.L Carrera Villacrés, Z. Li, Z. Zhao, T. Braun, and A. Neto, "A real-time indoor tracking system in smartphones," *MSWiM '16 Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 292–301, November 2016.

J.L Carrera Villacrés, Z. Li, Z. Zhao, and T. Braun, "A real-time indoor tracking system by fusing inertial sensor, radio signal and floor plan," *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, October 2016.